LE PROBLEME DE LA RECHERCHE D'UN FLOT D'EQUILIBRE SUR UN RESEAU ROUTIER

par

Clermont Dupuis

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophie Doctor (Ph.D.)

Cette thèse a été publiée grâce à une subvention du Fonds F.C.A.C. pour l'aide et le soutien à la recherche.

Université de Montréal

Centre de recherche sur les transports - Publication #344

Décembre 1983

Dépôt légal - 2e trimestre 1984

RECONNAISSANCE

Je voudrais d'abord exprimer ma gratitude envers mon directeur de recherches Monsieur Sang Nguyen, professeur au département d'informatique et de recherche opérationnelle de l'Université de Montréal, pour ses conseils et sa confiance, témoignés tout au long de ces dernières années.

Je désire également remercier tous mes amis et collègues pour les nombreuses discussions fructueuses que nous avons eues et pour avoir partagé mes joies, mes espérances et mes désespoirs...

L'aide financière et le support technique qui m'ont été accordés par le Conseil national de recherches du Canada, le ministère de l'Education du Québec et le Centre de recherche sur les Transports de l'Université de Montréal furent très appréciés.

Je tiens aussi à souligner le magnifique travail de dactylographie de Madame C. Marcil et de Madame Mâsse. Leur patience et leur doigté m'ont été d'un grand secours.

En terminant, je voudrais décerner une mention spéciale à Claudine, pour son encouragement et sa compréhension. Merci du fond du coeur...

TABLE DES MATIERES

	Page
Reconnaissance	ii
Table des matières	v
Table des illustrations	ix
Résumé	xiii
Introduction	1
CHAPITRE 1 : Problème d'équilibre de trafic	11
1.1 Concepts d'équilibre	11
1.2 Formulations mathématiques	18
1.2.1 Terminologie et notation	18
1.2.2 Formulations mathématiques	19
1.3 Une formulation duale à celle de Smith (1979)	21
1.4 Existence et unicité d'une solution d'équilibre	30
CHAPITRE 2 : Recherche d'un point d'équilibre lorsque la demande est fixe	37
2.1 Problème de Beckmann: coûts symétriques	37
2.2 Problème général: coûts asymétriques	38
2.2.1 Introduction aux méthodes de coupe	38
2.2.2 Méthode d'accumulation des contraintes	41
2.2.3 Taux de convergence de la méthode d'accumulation des contraintes	52
2.2.4 Parallèle avec l'algorithme de Frank et Wolfe	57
CHAPITRE 3 : Méthode d'accumulation des contraintes: résultats des calculs	61
3.1 Introduction	61
3.2 Implantation de l'algorithme: méthode de Dantzig-Wolfe	64
3.3 Structures de donnée, espace-mémoire et modules de programmation entrant en jeu	69
3.4 Résultats des calculs sur les réseaux de Hull et de Winnipeg	72
3.5 Techniques d'accélération de convergence de la méthode d'accumulation des contraintes	89
3.6 Comparaison avec d'autres algorithmes déjà existants	94

Page

CHAPITRE 4 : Lien avec la programmation semi-infinie..... 117

4.1 Introduction..... 117

4.2 Formulation et résolution du problème d'équilibre comme un programme linéaire semi-infini..... 118

CHAPITRE 5 : Approximation extérieure de l'ensemble des flots d'équilibre. 123

5.1 Introduction..... 123

5.2 Description d'une classe d'algorithmes pour le calcul d'un point d'équilibre..... 123

CHAPITRE 6 : Approche générale de calcul d'un flot d'équilibre dans le cadre de la programmation convexe..... 129

6.1 Introduction..... 129

6.2 Une méthode basée sur la représentation barycentrique des convexes.... 130

6.3 Avenues possibles..... 133

CHAPITRE 7 : Méthodes de points fixes, de projections, de sous-gradients.. 135

7.1 Introduction..... 135

7.2 Survol des méthodes de projection pour résoudre un problème d'équilibre général..... 136

7.3 Méthodes du sous-gradient..... 142

7.4 Généralisation des méthodes de projection pour résoudre un problème d'équilibre général: cas du flot sur les arcs..... 157

7.5 Essais effectués à partir d'un exemple de Dafermos (1980 a)..... 168

7.6 Généralisation des méthodes de projection pour résoudre un problème d'équilibre général: cas du flot sur les chemins..... 178

CHAPITRE 8 : Extension au cas de la demande élastique..... 187

8.1 Introduction..... 187

8.2 Méthode d'accumulation des contraintes..... 188

8.3 Adaptation de la méthode de Benders généralisée..... 193

Page

CHAPITRE 9 : Résolution d'un problème général de programmation convexe à l'aide d'une formulation primale-duale..... 199

9.1 Une formulation primale-duale d'un programme convexe..... 199

9.2 Une nouvelle approche de résolution d'un problème de programmation convexe..... 214

9.3 Cas spécial d'un problème de programmation convexe non-différentiable partout..... 217

Conclusion..... 221

Appendice A - Essai sur un réseau de petite taille [Nguyen, Dupuis, 1981] 223

Appendice B - Un exemple de Steenbrink (1974)..... 233

Appendice C - Programme FORTRAN qui permet de déterminer le flot d'équilibre sur le réseau de Hull grâce à la méthode d'accumulation des contraintes..... 235

Bibliographie..... 267

TABLE DES ILLUSTRATIONS

Page

Figure 1.1.1 : Configuration du réseau utilisé dans les 3 exemples de Aashtiani (79)..... 15

Figure 1.1.2 : Coûts de parcours associés à l'exemple 1.1.1..... 15

Figure 1.1.3 : Coûts de parcours associés à l'exemple 1.1.2..... 15

Figure 1.1.4 : Coûts de parcours associés à l'exemple 1.1.3..... 16

Graphique 1.3.1: Représentation de la fonction objectif du problème MAX-MIN dans l'exemple 1.3.1..... 24

Graphique 1.3.2: Représentation de la fonction objectif du problème MAX-MIN dans l'exemple 1.3.2..... 25

Graphique 1.4.1: Représentation de la fonction objectif du problème d'optimum collectif dans l'exemple 1.3.1..... 35

Tableau 3.4.1 : Résultats obtenus avec le réseau de Hull à partir de la méthode d'accumulation des contraintes (résolution approximative de chaque sous-problème linéaire): cas symétrique 75

Graphique 3.4.1: La norme $\|x^k - \bar{x}\|$ vs le nombre d'itérations k..... 77

Tableau 3.4.2 : Résultats obtenus avec le réseau de Winnipeg à partir de la méthode d'accumulation des contraintes (résolution approximative de chaque sous-problème linéaire): cas symétrique..... 79

Tableau 3.4.3 : Résultats obtenus avec le réseau de Hull à partir de la méthode d'accumulation des contraintes (résolution exacte de chaque sous-problème linéaire): cas asymétrique..... 81

Tableau 3.4.4 : Résultats obtenus avec le réseau de Hull (cas asymétrique) à partir d'une heuristique qui consiste à résoudre approximativement chaque sous-problème linéaire..... 82

Tableau 3.4.5 : Résolution exacte du sous-problème linéaire et rejet des contraintes non-actives aux itérations 1, 6, 12, 18,... (réseau de Hull: cas asymétrique)..... 83

Graphique 3.4.2: Courbe de Gap_k vs k (no. de l'itération) d'après les résultats du tableau 3.4.4..... 84

Page

Tableau 3.4.6 : Résultats obtenus en appliquant l'algorithme 3.4.1: le rejet des contraintes non-actives est relié à la décroissance du "GAP standard" (réseau de Hull, cas asymétrique) 88

Tableau 3.6.1 : Algorithme de Zuhovickii et al. appliqué au réseau de Hull (cas symétrique)..... 96

Tableau 3.6.2 : Résultats obtenus avec le réseau de Hull en utilisant un pas prédéterminé (cas symétrique)..... 98

Tableau 3.6.3 : Résultats obtenus avec le réseau de Hull à partir de l'algorithme de Frank et Wolfe (cas symétrique)..... 99

Graphique 3.6.1: Comparaison entre l'algorithme de Frank & Wolfe, la méthode d'accumulation des contraintes et celle avec un pas fixe (réseau de Hull, cas symétrique)..... 100

Graphique 3.6.2: Comparaison entre l'approche de PARTAN, celle de restriction et la méthode d'accumulation des contraintes (réseau de Hull, cas symétrique)..... 101

Tableau 3.6.4 : Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 10.0/t$) pour le réseau de Hull (cas symétrique).... 105

Tableau 3.6.5 : Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 100.0/t$) pour le réseau de Hull (cas symétrique)... 106

Graphique 3.6.3: Comparaison entre l'algorithme de Frank & Wolfe, la méthode d'accumulation des contraintes et une variante de l'algorithme de Hearn (cas symétrique)..... 107

Tableau 3.6.6 : Algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 10.0/t$) où chaque sous-problème d'équilibre est résolu par la méthode d'accumulation des contraintes (réseau de Hull, cas symétrique)..... 108

Graphique 3.6.4: Comparaison entre l'algorithme de Frank & Wolfe, l'approche de PARTAN et la méthode d'accumulation des contraintes (Réseau de Winnipeg, cas symétrique)..... 109

Tableau 3.6.7 : Résolution du problème <asymétrique> à partir de la méthode de diagonalisation (la solution x^k avec coûts symétriques diagonalisés est obtenue après 50 itérations de la méthode d'accumulation des contraintes)..... 111

Page

Tableau 3.6.8 : Résolution du problème <<asymétrique>> à partir de la méthode de diagonalisation (la solution x^k avec coûts symétriques diagonalisés est obtenue après 100 itérations de la méthode d'accumulation des contraintes..... 112

Tableau 3.6.9 : Résolution du problème <<asymétrique>> à partir d'une méthode de projection (la solution x^k avec coûts <<symétriques>> diagonalisés est obtenue après 50 itérations de la méthode d'accumulation des contraintes..... 113

Tableau 3.6.10 : Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 10.0/t$) (cas asymétrique)..... 114

Tableau 3.6.11 : Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 100.0/t$) (cas asymétrique)..... 115

Figure 7.5.1 : Réseau de Dafermos (1980 a)..... 168

Tableau 7.5.1 : Algorithme de projection de la section 7.4..... 172

Tableau 7.5.2 : Variante de l'algorithme de projection de la section 7.4. 173

Tableau 7.5.3 : Algorithme proposé par Sibony à la section 7.2..... 175

Tableau 7.5.4 : Méthode de sous-gradient de Polyak..... 176

Graphique 9.1.1: Représentation de l'ensemble-solution des problèmes associés à l'exemple 9.1.1..... 207

Graphique 9.1.2: Représentation de l'ensemble-solution des problèmes associés à l'exemple 9.1.2..... 208

Graphique 9.1.3: Représentation de l'ensemble-solution des problèmes associés à l'exemple 9.1.3..... 209

Figure A.1 : Réseau utilisé..... 225

Tableau A.1 : Matrice de demande..... 225

Tableau A.2 : Description des arcs et des coûts "symétriques"..... 226

Tableau A.3 : Résultats lorsque les coûts sont "symétriques"..... 227

Tableau A.4 : Description des coûts asymétriques (diagonale dominante). 228

Page

Tableau A.5 : Résultats obtenus lorsque les coûts sont "non-symétriques", affines et le Jacobien de la fonction de coût est diagonale dominant..... 229

Tableau A.6 : Description des coûts asymétriques (diagonale non-dominante) 230

Tableau A.7 : Résultats obtenus lorsque les coûts sont "asymétriques" et affines, et le Jacobien de la fonction de coût est non-diagonale dominante..... 230

Tableau A.8 : Résultats obtenus lorsque les coûts sont "asymétriques" et affines, et le Jacobien est diagonale dominant..... 231

Figure B.1 : Réseau de Steenbrink (1974)..... 233

Tableau B.1 : Matrice O/D de cet exemple..... 233

SOMMAIRE

Nous sommes concernés par le problème d'affectation du trafic routier suivant un modèle d'équilibre. Dans ce modèle, le coût de transport sur un tronçon de route est une fonction monotone du vecteur des flots des véhicules sur le réseau entier.

Etant donné une demande de déplacement entre certaines paires origine-destination du réseau, le problème d'affectation de trafic consiste à déterminer les chemins utilisés et la charge qui en résulte sur les arcs du réseau.

Dans le cas de trafic sans congestion importante, la grande majorité des automobilistes voyagent sur le chemin à coût de transport minimum reliant chaque paire origine-destination. Si le chemin à coût de transport minimum était l'unique chemin utilisé entre chaque paire origine-destination, la tâche serait aisée.

Lorsque la congestion prédomine et que la demande de trafic se répartit sur plusieurs routes entre chaque paire origine-destination, il devient difficile de déterminer d'une part les chemins utilisés et d'autre part la proportion de la demande empruntant chacun de ceux-ci.

En considérant une approche différente, où il n'est plus nécessaire de déterminer a priori les itinéraires et leur taux d'utilisation et, en faisant l'hypothèse que l'automobiliste choisit son itinéraire de façon à minimiser son coût personnel de transport, un facteur de compétition intervient, conduisant à un trafic d'équilibre dans lequel aucun automobiliste ne peut diminuer son coût de parcours en changeant unilatéralement son itinéraire.

Le calcul d'un flot d'équilibre consiste donc à rechercher la situation suivante: toutes les routes utilisées entre deux zones quelconques ont le même coût de parcours, tandis que les routes non-utilisées ont des coûts de parcours plus élevés [Wardrop, 1952].

Sans l'hypothèse standard de symétrie, le problème d'affectation de trafic est difficile. Plusieurs approches de résolution ont déjà été examinées par le passé; mais, malheureusement, elles ne s'avèrent pas complètement acceptables. En effet, elles exigent soit des conditions de convergence très sévères et/ou des quantités de calcul imposantes.

L'objectif global de cette thèse est orienté sur le développement de méthodes efficaces pour résoudre des problèmes de grande taille. En particulier, nous avons mis en relief une nouvelle formulation du problème d'équilibre. Celle-ci est montrée équivalente à la formulation variationnelle de Smith (1979) sous des conditions de monotonie de la fonction de coût.

Cette formulation nous a permis de développer une nouvelle méthode d'accumulation des contraintes uniquement sous des hypothèses de stricte monotonie et de continuité de la fonction de coût sur les arcs.

Nous avons analysé la convergence de l'algorithme et étudié différentes techniques d'accélération de convergence de notre méthode. Un logiciel efficace a été développé et divers essais sur des réseaux de moyennes et grandes tailles ont été effectués. Les résultats obtenus démontrent clairement l'intérêt de cette approche.

Afin de mieux comprendre les fondements sur lesquels repose notre méthode, nous avons fait le lien avec l'algorithme de Frank & Wolfe qui a été développé lorsque des hypothèses de symétrie sont imposées; puis, nous avons introduit la programmation semi-infinie et un concept d'approximation toujours plus raffinée de l'ensemble des flots d'équilibre.

En terminant, nous avons examiné différentes méthodes de points fixes, de projections et de sous-gradients qui découlent de notre formulation de l'équilibre. Puis, nous avons aussi envisagé deux extensions à notre modèle. La première est l'affectation de trafic d'équilibre avec demande élastique. La deuxième est l'application de notre approche à la résolution d'un problème convexe général.

INTRODUCTION

Le transport joue un rôle essentiel dans notre société. Chaque année, plusieurs décisions sont prises touchant ce domaine. Or, les nouvelles lois et la planification à court et moyen terme concernant le transport provoquent irrémédiablement des changements de comportement chez l'individu, des bouleversements dans notre mode de vie, voir même dans nos façons de pensée.

Selon Ben Barkow, psychologue [Leslie-Jeffery, 1979], lorsque nous demandons aux Canadiens ce qu'ils désirent en matière de transport, ils veulent sans aucun doute: "Un bon service, des billets faciles à déchiffrer, un système de tarifs cohérent, un minimum de surprises, des messages clairs, beaucoup d'espace, la possibilité de choisir ce qu'ils préfèrent et la certitude que l'on s'occupe de leur bien-être." Toutefois, les usagers d'un système de transport ne sont pas homogènes: ils n'expriment pas toujours leurs vrais besoins, leurs désirs et leurs inquiétudes.

Un objectif louable est de chercher à comprendre les préférences des gens puis, de tenter d'adapter les modes et les installations de transport à leurs besoins et préférences. Il est donc important de découvrir comment les gens réagissent, ce qu'ils font ou non. En effet, les planificateurs sont plus ou moins représentatifs de la population, de sorte que nous devons leur fournir des outils "scientifiques" leur permettant de légiférer de façon éclairée.

Un autre facteur qui touche les planificateurs en transport est la représentation du temps pour une personne qui se déplace. Les gens ignorent combien d'argent ils accepteraient en échange de leur tepms, à moins d'être placés dans une telle situation. Des études sur le sujet ont été faites par des chercheurs de l'Université de Toronto [Twidale, 1981]. Nous y sommes invités à réfléchir sur les avantages à éliminer la contrainte du temps en transport. En effet, on tente habituellement d'améliorer le système en place dans le but de gagner quelques minutes et cela au prix de sommes énormes, alors que l'on obtiendrait des résultats beaucoup plus probants en diminuant l'importance des retards et en généralisant la pratique des heures flexibles.

Bien d'autres phénomènes entrent en jeu. Nous pouvons penser à des contraintes budgétaires économiques et sociales, des données géographiques, la crise énergétique, les pollutions de toutes sortes, etc.

On admet facilement l'ampleur et la complexité du domaine des transports. Afin de développer des systèmes efficaces, acceptés par la population et compatibles avec l'environnement, la recherche a été axée sur le développement de modèles mathématiques sophistiqués permettant de mieux comprendre les problèmes et de planifier pour des besoins futurs.

En s'intéressant au monde du transport, nous sommes concernés par les préoccupations suivantes: l'économie des transports, la sécurité routière, le transport urbain, l'innovation technologique et l'économie de l'énergie en transport.

Par contre, nous nous restreindrons dans cette thèse au secteur du transport urbain. Strobel (1982) distingue cinq catégories de problèmes que l'on rencontre dans le transport urbain: la mobilité, la sécurité des usagers, l'environnement, les ressources en présence et l'efficacité de même que l'attrait du transport public.

Les problèmes liés à la mobilité en milieu urbain deviennent de plus en plus critiques et les choix offerts sont à des degrés divers, des facteurs déterminants sur l'environnement, la qualité de la vie et par conséquent, sur le bien-être de l'homme.

La congestion peut diminuer sensiblement la vitesse de parcours ainsi que le rapport # d'usagers/distance parcourue. Elle cause aussi une dispersion du temps de parcours rendant difficile l'estimation de ce facteur.

De plus, la congestion est très reliée au nombre d'accidents d'automobiles qui surviennent. Il semble que les programmes d'améliorations routières donnent de meilleurs résultats que toutes les campagnes de sécurité visant les conducteurs et les véhicules réunis. Les problèmes de pollution de l'air et du bruit souvent discutés provoquent une émigration de la population des centres villes vers les banlieux.

Le bruit engendré par la circulation devient de plus en plus envahissant dans le temps et l'espace; il devient pour l'individu un élément stressant et agressif. Ses effets, encore mal connus, sont d'ordre physiologique, psychologique, et socio-économique. De plus, la pollution de l'air et les dangers pour la santé de l'homme sont connus depuis longtemps.

Nos systèmes de transport consomment beaucoup d'énergie. Sous des conditions de congestion, cette consommation peut s'accroître significativement. Cela est dû à des manoeuvres d'accélérations et de décélérations continuelles. Selon un rapport fait en 1977, au Québec, les transports accaparent le quart de l'énergie, et plus du tiers des produits pétroliers. Dans la région de Montréal, la voiture particulière consomme à elle seule près des deux tiers de toute l'énergie consacrée au Québec aux déplacements urbains par automobile [Rapport du comité des transports de la région de Montréal, 1977].

En tenant compte aussi des problèmes de stationnement, nous pouvons nous demander à quel degré nous subissons l'attrait du transport en commun? Répond-il à nos besoins? L'automobile dit-on possède certains avantages, tels la fiabilité, la flexibilité, la sécurité, le confort, la rapidité, l'individualité, la plaisance, etc. La dépendance de certaines personnes face à l'automobile, les avantages qu'on lui attribue et la mentalité nord-américaine qui accorde encore beaucoup de prestiges à posséder une automobile, expliquent la prépondérance de l'utilisation de l'automobile au Québec [Recherches Transport, 1982].

Le planificateur en transport urbain a donc besoin d'outils mathématiques lui permettant d'acquérir une vision éclairée de la situation.

Malgré un élargissement important du réseau routier au cours des dernières décennies, la congestion de la circulation, à certaines heures et en différents points du réseau, persiste et va jusqu'à augmenter. Il semble que la solution à ce problème ne réside pas tant dans l'addition de nouvelles infrastructures routières - la capacité du réseau n'est pas épuisée - qu'en une utilisation plus rationnelle de ces dernières.

L'objectif premier de cette thèse est la prédiction des flots de véhicules dans un réseau congestionné, compte tenu du comportement des usagers. Nous utilisons à cette fin des modèles d'équilibre de trafic. Afin de préciser notre cadre de travail, examinons brièvement les différentes phases du processus de planification du transport et quel est le rôle des modèles de réseaux dans ce processus.

Potts & Oliver (1972) distinguent quatre phases dans le processus classique de la planification en transport urbain.

La première phase concerne la préparation des données. La région d'étude doit être définie; des divisions par secteurs et par zones sont effectuées. Cela nous amène alors à codifier un réseau routier (et/ou de transport en commun), à rechercher les caractéristiques du réseau c'est-à-dire, les capacités sur les arcs, le temps de parcours, les distances à parcourir, etc. Après cette collecte de données, nous devons déterminer les déplacements interzonaux. Cela est fait grâce à des enquêtes "origine-destination" et des comptages de trafic. Un dernier point consiste à se renseigner sur les plans d'utilisation du sol, les facteurs de croissance (ou de décroissance) des emplois, du commerce, de la population, etc.

Dans la deuxième phase, nous nous intéressons à la calibration des modèles. Nous pouvons identifier quatre types de modèles:

A) Le modèle de génération:

Il relie les facteurs socio-économiques à la production de déplacements interzonaux. Le modèle a pour tâche de déterminer le nombre total de voyages issus d'une origine ou arrivant à une destination. En général, des modèles de régression sont utilisés où les variables explicatives sont les surfaces commerciales, l'emploi, la population, les revenus, le taux de passage des véhicules, etc.

B) Le modèle de distribution:

Celui-ci produit une matrice de déplacements interzonaux. En considérant le coût généralisé de parcours entre chaque paire origine-destination (O-D) et le nombre total des voyages de chaque origine et de chaque destination, il s'agit d'estimer le nombre de voyages entre chaque paire O-D.

Nous retrouvons des modèles utilisés en économétrie, des modèles normatifs, des modèles de gravité, etc. (Potts & Oliver, 1972).

C) Le modèle de choix modal:

Il permet de distinguer entre les différents modes de transport.

D) Le modèle d'affectation du trafic:

Nous cherchons à répartir le trafic sur le réseau étant donné les demandes O-D. Nous verrons plus loin sur quels principes ce modèle est basé et les différents problèmes qui en résultent.

Remarquons l'existence de modèles intégrés distribution - affectation (Florian et al., 1975; Erlander, 1979; etc.).

Dans la troisième phase, l'objectif est de prévoir les déplacements futurs. Enfin, la dernière phase consiste à évaluer les réseaux en place.

Notre intérêt portera donc sur le modèle d'affectation du trafic où nous devrons déterminer la charge résultante sur chaque arc du réseau lorsque la demande pour chaque paire O-D est fixée.

Décrivons maintenant les éléments qui entrent en jeu dans ce modèle d'équilibre.

Nous examinons ici chacun des constituants de ce modèle. Mentionnons d'abord que les infrastructures du transport urbain sont représentées avantageusement par une structure mathématique appelée "réseau".

A n'importe quel flot de trafic sur le réseau, on peut associer différentes mesures de performance telles que le coût d'un voyage, le temps, le confort, etc. Ces composantes ont des répercussions importantes sur le comportement des usagers; toutefois, il est difficile de toutes les inclure dans un modèle (la sécurité, le confort, etc.). Habituellement, le temps et le coût de voyage sont seulement considérés.

Les usagers de ce réseau de transport constituent aussi un élément important de ce modèle. Ils devront choisir leur mode de transport. La route empruntée, leur destination, etc. De là intervient une fonction de demande qui donne le flot de trafic dans le système. Cette fonction dépend du réseau en présence et des mesures de performance. Le calibrage de celle-ci est une tâche difficile. Nous devons imposer des hypothèses supplémentaires, sans quoi cela est impossible (Aashtiani, 1979). Aashtiani mentionne par exemple un procédé appelé agrégation où l'on rassemble les individus selon leur revenu, leur lieu résidentiel, leur profession, etc. A l'intérieur d'un groupe, nous supposons que les individus répondent de façon semblable.

Nous pouvons considérer aussi que la population de chaque zone est homogène de sorte chaque zone est représentée par un point appelé "centroïde" sur le réseau.

Aashtiani indique encore un dernier type d'agrégation concernant le réseau lui-même, au niveau de ses sommets et de ses arcs.

A moins d'avis contraire, nous assumerons une demande constante telle que le processus d'agrégation est terminé. Nous supposerons de plus qu'aucun élément stochastique n'entre en jeu.

Décrivons maintenant le dernier élément du modèle, celui du temps d'attente et du temps de parcours sur les arcs du réseau. Nous supposons qu'il n'y a aucun délai sur les sommets car, autrement, le réseau peut toujours être modifié en conséquence. Le délai sur chaque arc peut dépendre du flot sur tous les arcs du réseau. Bien des modèles ont été construits sur la base que le délai sur un arc donné dépend uniquement du flot sur cet arc; pourtant, l'influence du flot sur les arcs voisins à une intersection par exemple, nous amène à penser le contraire. De plus, en considérant des fonctions de coût (délai) généralisées,

notre modèle peut tenir compte de plusieurs modes de transport et de classes d'usagers. Aashtiani (1979) donne une liste assez complète des formes générales des fonctions de coût que l'on rencontre fréquemment dans la littérature. Ces fonctions sont habituellement continues, monotones et positives.

Dans le chapitre suivant, nous allons examiner différents critères d'affectation de trafic et faire le lien avec la définition originale du concept d'équilibre égoïste ("User-Optimum") (Wardrop, 52). Le but visé à travers ces critères d'affectation de trafic est de refléter le mieux possible le comportement des usagers face à une situation de congestion et de formuler mathématiquement notre problème. Des conditions d'existence et d'unicité d'un trafic d'équilibre sont aussi données.

Notre intérêt portera surtout sur la formulation variationnelle de Smith (79). Nous proposerons une formulation complémentaire à celle de Smith (79), qui lui est équivalente sous des hypothèses de monotonie de la fonction de délai. Celle-ci peut dépendre du flot sur tout le réseau. Aucune hypothèse de symétrie sera imposée.

Dans le deuxième chapitre, grâce à ce nouveau critère d'affectation de trafic, nous avons pu développer une méthode originale d'accumulation des contraintes afin de déterminer un flot d'équilibre. Une analyse du taux de convergence et le parallèle avec l'algorithme de Frank & Wolfe (56) sont effectués. La formulation de Beckmann (56) y est aussi mise en évidence; elle prend un sens lorsque le Jacobien de la fonction de délai est symétrique.

L'intérêt de cet algorithme est sans équivoque sachant que les méthodes de recherche d'un flot d'équilibre égoïste préconisées jusqu'à date pour des réseaux de grande taille exigent soient des temps de calcul et/ou de l'espace-mémoire considérables, soient des hypothèses trop restrictives pour s'assurer la convergence. L'objectif premier de ce rapport est donc de remédier à cette situation.

Au chapitre trois, nous décrivons en détail l'implantation de notre méthode, les raisons qui nous ont fait choisir une approche primale de Dantzig-Wolfe et les différentes alternatives qui s'offrent à nous. Plusieurs techniques d'accélération de convergence et d'économie de mémoire sont étudiées. Des essais sur ordinateur ont été effectués pour des réseaux de taille moyenne. Plusieurs comparaisons ont été faites avec des algorithmes existants et des variantes de ceux-ci.

Nous avons considéré aussi lors de l'implantation de notre méthode de coupe, une approche duale faisant intervenir successivement l'algorithme primal du simplexe, l'algorithme dual du simplexe et une technique de relaxation.

Dans les chapitres subséquents, nous avons fait le lien entre notre méthode d'accumulation des contraintes et la programmation semi-infinie.

De plus, nous avons élargi notre cadre théorique en considérant notre approche comme une approximation extérieure de plus en plus raffinée de l'ensemble convexe, compact des flots d'équilibre (égoïste). Dès lors, nous avons construit une classe d'algorithmes pour le calcul d'un point d'équilibre englobant la méthode d'accumulation des contraintes. Cela nous a permis de faire le parallèle avec la méthode de diagonalisation [Bensoussan et al., 72; Dafermos, 80; Florian, 81].

En imposant des hypothèses supplémentaires sur les fonctions de coût comme par exemple, la forte monotonie, différentes méthodes de points fixes, de sous-gradients, des algorithmes pour calculer un point de selle, etc. ont été développés dans le but d'obtenir un optimum-égoïste.

En terminant, nous proposons deux extensions possibles. La première consiste à traiter le cas général où la demande est élastique; différents algorithmes et heuristiques seront étudiés dans cette optique.

En second lieu, nous allons examiner un problème de programmation convexe général. Des conditions nécessaires et suffisantes d'optimalité pour un tel problème [Kuhn et Tucker,61] permettent l'élaboration de plusieurs approches de résolution. Ces méthodes de résolution se distinguent essentiellement selon le type d'approche utilisé, soit primal, dual ou primal-dual. Les méthodes primales génèrent des points réalisables du problème tout en s'approchant graduellement de l'optimum selon un certain critère au choix. L'approche duale consiste à avancer progressivement vers l'optimum de l'extérieur de l'ensemble admissible. Malheureusement, elle ne nous permet pas d'obtenir directement les variables primales correspondantes de façon générale. Enfin, l'approche primale-duale s'intéresse à la fois aux deux problèmes primal et dual afin de satisfaire les conditions d'optimalité. Nous considérons dans ce dernier chapitre une nouvelle formulation duale faisant intervenir les variables primales seulement. Celle-ci peut s'écrire comme un problème MAX-MIN où l'objectif "externe" est concave mais non-nécessairement différentiable. De cette façon, nous avons pu appliquer notre méthode d'accumulation des contraintes pour résoudre le problème dual et par conséquent, le problème initial. D'autres algorithmes ont aussi été étudiés brièvement pour certaines sous-classes de problèmes.

CHAPITRE 1

PROBLEME D'EQUILIBRE DE TRAFIC

1.1 CONCEPTS D'EQUILIBRE

Nous examinerons en détail deux concepts d'équilibre et tâcherons d'expliciter davantage les liens qui existent entre les différentes formulations qui en découlent.

Le comportement des usagers du transport urbain est difficile à prévoir. Plusieurs facteurs entrent en jeu: le temps de parcours, l'état de la route, le confort, le nombre d'intersections, etc. Par conséquent, l'hypothèse simplificatrice la plus logique est telle que l'usager choisit son itinéraire de sorte que son coût de parcours soit minimisé. Cela entraîne que toutes les routes utilisées entre une paire origine-destination ont le même coût de parcours, tandis que les routes non-utilisées possèdent un coût de parcours plus élevé. Nous pouvons alors énoncer le premier principe de Wardrop (1952):

> Pour toute paire O-D, il ne doit exister aucun chemin de coût de transport inférieur à celui d'un chemin de flot positif. (1.1.1)

Cette notion d'équilibre est souvent identifiée sous le vocable d'"équilibre-égoïste" (User-optimum).

Wardrop a énoncé ce principe afin de formuler mathématiquement le problème; cependant, à l'origine, il avait plutôt proposé la notion d'équilibre suivante:

> A l'équilibre, aucun usager ne peut améliorer son temps de parcours en changeant unilatéralement de chemins. (1.1.2)

Aashtiani (1979) donne trois situations où les deux principes précédents ne sont pas équivalents: variables de flots discrètes, des fonctions de coûts sur les arcs non-continues, des fonctions de coûts sur les arcs non-croissantes.

Nous allons aussi considérer le principe suivant:

A l'équilibre, les usagers ne peuvent changer de parcours tout en améliorant le coût total de parcours basé sur le coût associé au nouveau parcours. (1.1.3)

Celui-ci découle d'un principe dynamique énoncé par Smith (1979):

Les usagers changent seulement de parcours lorsque le coût total de parcours, basé sur le coût des routes d'aujourd'hui, est réduit. (1.1.4)

Comme Smith (1979) nous le fait remarquer, en considérant le principe (1.1.2), à l'équilibre, aucun usager ne peut réduire son temps de parcours en changeant unilatéralement de chemins de sorte que le coût total de parcours (basé sur le coût de tout nouveau parcours) ne peut être amélioré. Par contre, les usagers obéissant au principe énoncé en (1.1.3) peuvent commettre une erreur et choisir une route plus coûteuse.

De même, une solution d'équilibre selon (1.1.2) l'est aussi selon (1.1.1).

Posons

$C_r(H) \equiv$ Le coût du parcours r lorsque le flot est H.

Nous pouvons alors formuler mathématiquement les principes (1.1.1) et (1.1.3) comme suit:

$$(1.1.1) \equiv c_r(H^*) > c_s(H^*) \Rightarrow H_r^* = 0, \qquad \forall r \neq s$$

où H_r^* désigne le flot du parcours r;

$$(1.1.3) \equiv c(F)\ (F-H) \geq 0 \text{ pour tout F réalisable.}$$

A ce stade-ci, il serait bon de voir à l'aide d'exemples les différences entre chacun de ces principes.

Dans un premier temps, nous examinerons les trois exemples donnés par Aashtiani (1979). Il considère le cas d'une seule paire O-D avec deux unités de flots et deux chemins (voir figure 1.1.1).

Exemple 1.1.1 - Variables entières

Les coûts de parcours sont donnés à la figure 1.1.2. Aashtiani constate que les fonctions de coût sont continues et non-décroissantes; mais, en considérant les variables de flot entières, il obtient:

Principe (1.1.1) : aucune solution d'équilibre
Principe (1.1.2) : $\{(1,1)\}$.

Or, en appliquant le principe (1.1.3), nous avons $\{(1,1),(2,0)\}$. Nous remarquons ainsi que dans le cas où les variables de flots sont entières, les solutions d'équilibre obéissant au principe (1.1.3) n'obéissent pas nécessairement au principe (1.1.1).

Exemple 1.1.2 - Fonctions de coût non-continues

(Voir la figure 1.1.3 pour une description des fonctions de coût)

Nous supposons maintenant que les variables de flots sont continues. Les principes (1.1.1) et (1.1.2) n'ont pas de solutions. Le principe (1.1.3) possède une solution unique $\{(1,1)\}$.

Exemple 1.1.3 - Fonction de coût décroissante

(Voir la figure 1.1.4 pour une description des fonctions de coût)

Nous obtenons:

Principe (1.1.1) : $\{(\lambda,2-\lambda)\,|\,\lambda\in[0,2]\}$
Principe (1.1.2) : $\{(0,2)\}$
Principe (1.1.3) : $\{(\lambda,2-\lambda)\,|\,\lambda\in[0,2]\}$.

Nous verrons plus loin que lorsque les fonctions de coûts et les variables de flots sont continues, les solutions obéissant au principe (1.1.3) satisfont aussi au principe (1.1.1).

Même sous des hypothèses de continuité et de non-décroissance des fonctions de coût, il n'existe pas de preuve formelle que les principes (1.1.1) et (1.1.2) soient équivalents. Par contre, sous des hypothèses d'hémi-continuité et de monotonicité des fonctions de coût, nous verrons que les principes (1.1.1) et (1.1.3) sont équivalents.

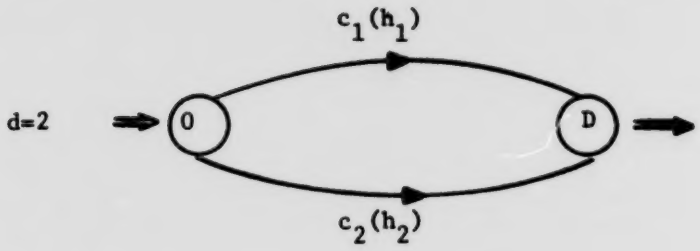


Figure 1.1.1 - Configuration du réseau utilisé dans les 3 exemples de Aashtiani (79)

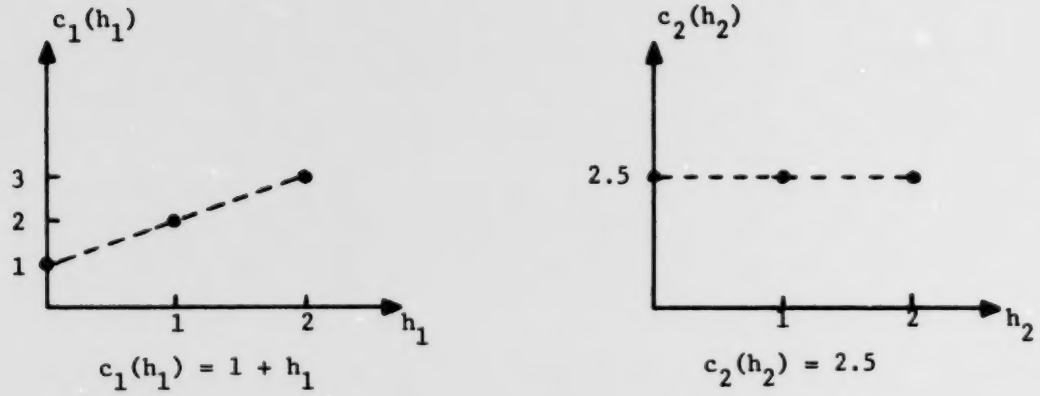


Figure 1.1.2 - Coûts de parcours associés à l'exemple 1.1.1.

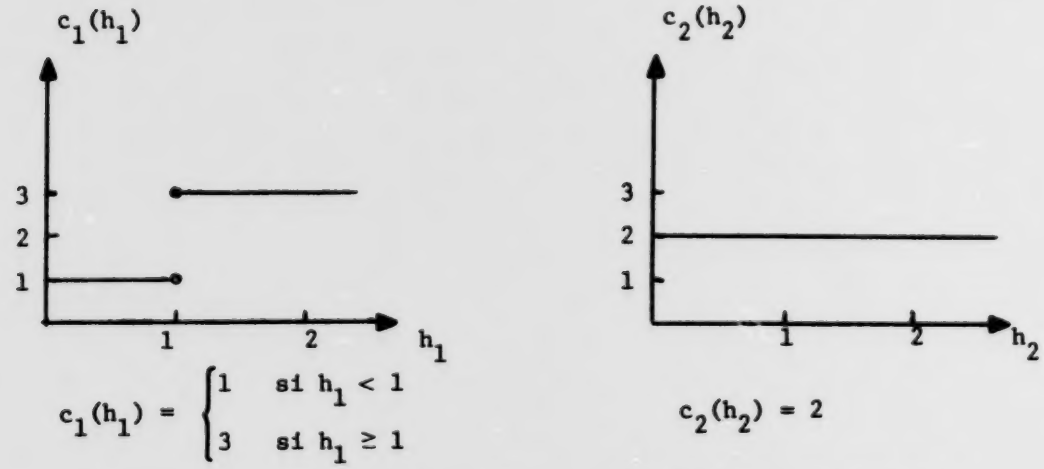
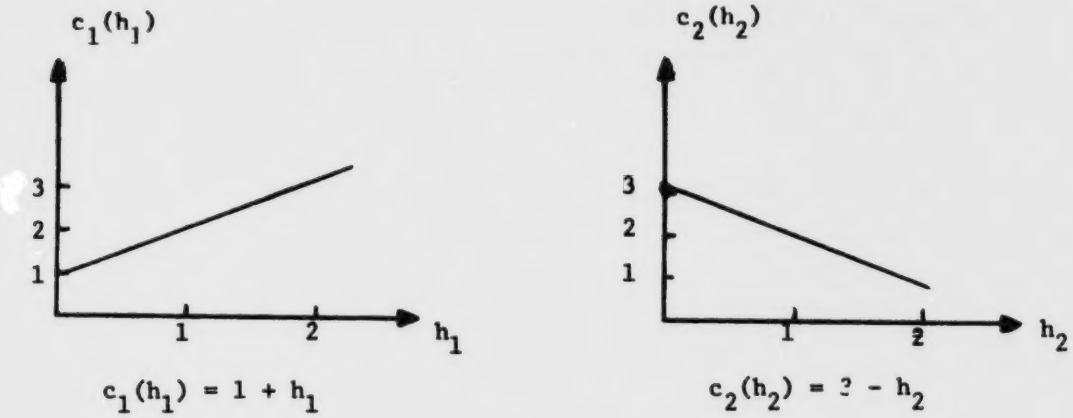


Figure 1.1.3 - Coûts de parcours associés à l'exemple 1.1.2.



$c_1(h_1) = 1 + h_1$ $\qquad$ $c_2(h_2) = 3 - h_2$

Figure 1.1.4: Coûts de parcours associés à l'exemple 1.1.3.

Comparons maintenant les principes (1.1.2) et (1.1.3). A l'aide des exemples précédents, nous nous apercevons que l'hypothèse de continuité des fonctions de coût et des variables de flot ainsi que la non-décroissance des fonctions de coût sont des conditions nécessaires d'équivalence entre ces deux principes. Pour mieux comprendre encore les liens qui existent entre les principes (1.1.2) et (1.1.3), nous nous attarderons aux trois exemples énoncés au chapitre 9. Chacun de ces exemples possède des fonctions de coût continues mais non-monotones, égales au vecteur gradient de la fonction objectif concernée (sauf dans l'exemple 9.1.3). Les variables de flot sont continues.

L'exemple (9.1.1) est un cas où les trois formulations sont équivalentes. Par contre, dans l'exemple (9.1.2), on obtient:

Principe (1.1.1): $\{(2,0,0)\} \cup \{(0,\lambda,2-\lambda) | \lambda \in [0,2]\}$
Principe (1.1.2): $\{(0,\lambda,2-\lambda) | \lambda \in [0,2]\}$
Principe (1.1.3) $\equiv$ Principe (1.1.2).

Enfin, considérons l'exemple (9.1.3), nous avons les résultats suivants:

Principe (1.1.1): $\{(0,\lambda,2-\lambda) | \lambda \in [0,1]\} \cup \{(\lambda,1,1-\lambda) | \lambda \in [0,1]\}$
Principe (1.1.2): aucune solution
Principe (1.1.3) $\equiv$ Principe (1.1.2).

A la lumière des exemples précédents, il semble que le principe énoncé en (1.1.2) soit très relié à la formulation duale (1.1.3). De plus, les solutions satisfaisant le principe (1.1.2) obéissent aussi au principe (1.1.3).

Smith (1979) fait intervenir une notion de stabilité très proche du principe (1.1.3).

Dans les prochains chapitres, nous utiliserons les différentes formulations qui découlent des principes (1.1.1) et (1.1.3) pour rechercher un point d'équilibre dans un réseau routier.

Nous introduirons maintenant un deuxième concept d'équilibre.

Le flot d'équilibre est caractérisé par une utilisation restreinte des infrastructures routières existantes. En effet, les usagers considérant uniquement leur coût personnel, sont en compétition pour obtenir le chemin à coût minimal. D'où, ils engendrent du même coup des congestions importantes. Un objectif commun serait de régir le trafic par un gouvernement central, où le choix des usagers serait fixé de façon à utiliser efficacement les infrastructures existantes. Cette situation correspond au deuxième principe de Wardrop (1952). Mathématiquement, il correspond à minimiser le coût total du système. Cet équilibre est souvent appelé "Optimum collectif (system optimum)".

Ce problème sera aussi examiné ultérieurement.

Nous donnons maintenant la notation utilisée et une description du problème.

1.2 FORMULATIONS MATHEMATIQUES

1.2.1 Terminologie et notation

On se donne un réseau de transport G formé de N sommets et M arcs. On distingue P paires origine-destination (O-D) et l'on dénote par d_p la demande de flot entre la pième paire O/D. Soit $x = (x_1, x_2, \ldots, x_M)$ le vecteur de flot où x_i est le flot sur l'arc i. $x \geq 0$ est un flot réalisable s'il satisfait les équations de conservation de flot habituelles. On dénotera $\Omega \subset \mathbb{R}^M$ l'ensemble des flots réalisables, on suppose Ω borné et non-vide (d_p fini, p = 1,2,... et il existe au moins un chemin reliant la pième paire O/D).

A chaque arc i du réseau est associé un coût de transport $C_i(x)$ fonction continue à valeurs réelles définie sur Ω. On désigne par C(x) la fonctionnelle $(C_1(x), C_2(x), \ldots, C_M(x))$: $C: \mathbb{R}^M \to \mathbb{R}^M$. Remarquons que la fonctionnelle C(x) est telle que chacun de ses éléments peut dépendre du flot sur chaque arc du réseau. De plus, le coût sur un chemin est égale à la somme des coûts sur chaque arc de ce chemin.

Nous avons supposé la continuité des fonctions de coût; toutefois, cette hypothèse peut être relaxée dans bien des situations.

Pour désigner un flot d'équilibre, nous utiliserons la notation $x^* \in \Omega$. Le contexte nous permettra facilement d'identifier le principe accolé à la notion d'équilibre (voir la section 1.1).

1.2.2 Formulations mathématiques

Considérons en premier lieu la notion d'équilibre suivante:

> Un flot x^* est un flot d'équilibre si et seulement si pour toute paire O/D, aucun chemin ne peut avoir un coût de transport strictement inférieur à celui d'un chemin portant un flot positif. (1.2.2.1)

Dans le cas simple où les coûts de transport sont monotones et "symétriques" (c'est-à-dire, le Jacobien de la fonction de coût est symétrique), la recherche d'un flot d'équilibre se ramène à la résolution d'un problème convexe [Beckmann et al., 1956]. Nous examinerons brièvement cette situation dans un chapitre ultérieur. Plusieurs méthodes de résolution relativement efficaces ont déjà été élaborées [Nguyen, 1976; Leblanc et al., 1975].

Dans des situations réelles, cette hypothèse de symétrie est rarement vérifiée, malheureusement. En général, l'équilibre de Wardrop peut être décrit mathématiquement par une inégalité variationnelle [Smith, 1979], c'est-à-dire, x^* est un flot d'équilibre si et seulement si

$$x^* \in \Omega, \quad (C(x^*), x - x^*) \geq 0 \quad \text{pour tout } x \in \Omega \tag{1.2.2.2}$$

où

(,) est le produit scalaire ordinaire dans R^M.

Bien qu'il existe déjà de nombreuses techniques de calcul numériques des inéquations variationnelles [Glowinski et al., 1976], le calcul d'un flot d'équilibre demeure très ardu. C'est pourquoi, nous présentons dans ce travail une nouvelle formulation du problème d'équilibre [Nguyen & Dupuis, 1981] qui se trouve être en quelque sorte le dual de la formulation de Smith.

Fisk & Nguyen (1980) ont suggéré un système d'équations non-linéaires pour décrire le problème d'équilibre. Ils ont considéré une classe de modèles où la demande est élastique. Ils proposent alors une méthode d'approximations successives, une méthode de Gauss-Seidel non-linéaire, une méthode de Jacobi non-linéaire et bien des variantes (voir Ortega & Rheinboldt, 1970). Toutefois, il s'avère difficile de vérifier les conditions de convergence de ces algorithmes. De plus, l'inconvénient majeur de cette méthode est la résolution à chaque itération d'un problème d'équilibre standard (avec coûts "symétriques").

La méthode proposée par Florian (1977) appartient à cette classe d'algorithmes; elle s'est avérée efficace pour ce modèle particulier [Fisk & Nguyen, 1980a].

Une dernière formulation consiste à représenter l'équilibre de Wardrop comme un système de complémentarité non-linéaire [Aashtiani et Magnanti, 1980]. Toutefois, cela n'a pas donné de méthodes de résolution adéquates pour des problèmes de grande taille.

Par conséquent, il n'est point exagéré d'affirmer qu'aucune méthode existante de calcul de flot d'équilibre sur des réseaux de grande taille ne soit complètement acceptable. Nous voulons proposer des algorithmes simples, n'exigeant pas trop de calculs à chaque itération, et dont les conditions de convergence sont faciles à vérifier et ne sont pas trop restrictives.

En second lieu, nous allons considérer le deuxième principe de Wardrop qui s'énonce comme suit:

> Le trafic se répartit sur le réseau de telle manière que le coût total moyen de parcours est minimum. (1.2.2.3)

Ce principe est formulé mathématiquement ainsi:

$$\text{Min } (C(x), x) \qquad (1.2.2.4)$$
$$\text{sujet à } x \in \Omega \ .$$

Nous constatons immédiatement que les conditions de Kuhn-Tucker de ce problème sont celles d'un problème d'équilibre obéissant au principe (1.2.2.1), où les coûts sont de la forme

$$\overline{C} = C + \pi$$

où

$$\pi_a = \sum_b f_b \partial c_b / \partial x_a$$

La fonction de coût $\overline{C}$ est symétrique.

Par conséquent, la résolution de (1.2.2.4) peut se ramener à celle d'un problème d'optimum-égoïste de sorte que les outils développés pour ce dernier sont aussi valables pour (1.2.2.4).

Nous n'appuierons pas davantage sur ce résultat. Nous nous attarderons plutôt sur le fait que

$$(C(x),\ x)$$

peut être strictement convexe même en l'absence de la monotonie de C(x) [Fisk et Nguyen, 1980B]. Dans un prochain chapitre, nous allons développer une méthode d'accumulation de contraintes pour résoudre le problème (1.2.2.4), à la lumière des résultats généraux du chapitre 2.

1.3 UNE FORMULATION DUALE A CELLE DE SMITH (1979)

Nous proposons dans ce document une nouvelle formulation au problème de la recherche d'un flot d'équilibre (1er principe de Wardrop) dans un réseau routier. Cette formulation s'obtient directement du principe énoncé en (1.1.3). Nous avons:

"Le trafic x^* est un flot d'équilibre selon le principe énoncé en (1.1.3)
si et seulement si
$x^* \in \Omega$, $(C(x), x-x^*) \geq 0$ pour tout $x \in \Omega$."

Examinons maintenant les liens qui existent entre cette formulation et l'inégalité variationnelle de Smith:

Proposition 1.3.1:

Soient

a) $x^* \in \Omega$, $(C(x^*), x-x^*) \geq 0$ pour tout $x \in \Omega$

b) $x^* \in \Omega$, $(C(x), x-x^*) \geq 0$ pour tout $x \in \Omega$

Alors

1) si la fonction de coût est hémi-continue sur Ω, alors chaque solution de b) est solution de a).
2) si de plus, la fonction de coût est monotone sur Ω, alors chaque solution de a) est solution de b).

Preuve: Elle est basée sur une propriété connue des opérateurs définis sur un ensemble convexe (voir par exemple Glowinski et al., 1976; Auslender, 1976).

La monotonie est une condition suffisante mais non nécessaire pour s'assurer l'équivalence des relations a) et b) dans la proposition (1.3.1) (voir les trois exemples du chapitre 9). Malheureusement, nous n'avons pu obtenir de conditions plus souples que celles-là.

Reformulons maintenant les relations a) et b) précédentes. L'inégalité variationnelle de Smith peut s'écrire:

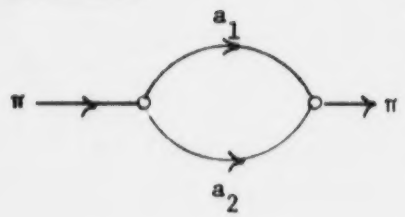
$$\underset{x \in \Omega}{\text{Max}}\ f(x) \qquad (1.3.1)$$

où

$$f(x) = \underset{y \in \Omega}{\text{Min}}\ (C(x), y-x) \qquad (1.3.2)$$

De façon générale, la fonction f n'est ni concave, ni différentiable. Considérons chacune des situations suivantes:

Exemple 1.3.1



$$c_{a_1} = -\cos x_{a_1}$$

$$c_{a_2} = 2$$

Le Jacobien de la fonction de coût donne:

$$\begin{bmatrix} \sin x_{a_1} & 0 \\ 0 & 0 \end{bmatrix}$$

une matrice symétrique, semi-définie positive $\forall x_{a_1} \in [0,\pi]$. D'où la fonction de coût est monotone sur l'ensemble des flots réalisables. La solution d'équilibre de ce problème est $\binom{\pi}{0}$.

Considérons maintenant la fonction objectif du problème (1.3.2) où $y_{a_1} = 0$ et $y_{a_2} = \pi$:

$$\begin{pmatrix} x_{a_1} \\ x_{a_2} \end{pmatrix} \longmapsto \begin{bmatrix} -\cos x_{a_1} \\ 2 \end{bmatrix}^t \quad \begin{bmatrix} -x_{a_1} \\ -x_{a_2} + \pi \end{bmatrix} .$$

La matrice hessienne de celle-ci s'écrit:

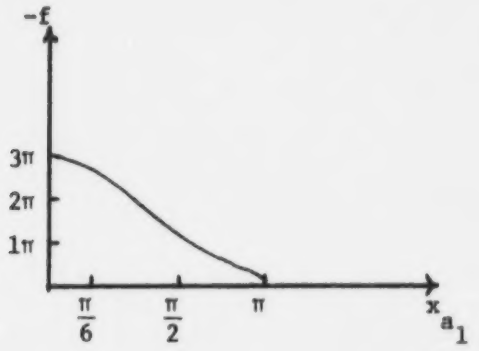
$$\begin{bmatrix} -2\sin x_{a_1} - x_{a_1} \cos x_{a_1} & 0 \\ 0 & 0 \end{bmatrix} .$$

Or, cette matrice possède une valeur propre positive lorsque x_{a_1} est près de π. Par conséquent, l'objectif du problème (1.3.2) n'est pas concave. Considérons maintenant la fonction $-f(x_{a_1}, x_{a_2})$ que l'on peut expliciter comme suit:

$$-\cos x_{a_1} (x_{a_1} - \pi) + 2x_{a_2} .$$

Graphiquement, nous obtenons:

Graphique 1.3.1 Représentation de la fonction objectif du problème MAX-MIN dans l'exemple 1.3.1.



La matrice hessienne de -f est:

$$\begin{bmatrix} (x_{a_1} - \pi) \cos x_{a_1} + 2\sin x_{a_1} & 0 \\ 0 & 0 \end{bmatrix}$$

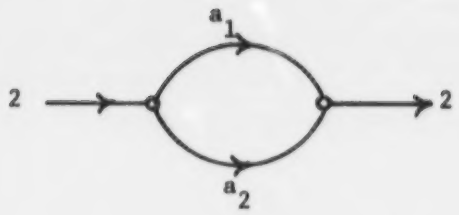
Celle-ci possède comme valeurs propres $-\pi$ et 0 lorsque $x_{a_1} = 0$. Par conséquent, la fonction f n'est pas concave dans cet exemple.

Nous nous sommes basés sur l'exemple donné par Hearn & Nguyen (1982) dans un cadre général pour construire un exemple dans le domaine des flots d'équilibre.

Considérons aussi une situation où la fonction f n'est pas différentiable partout.

Exemple 1.3.2

Etant donné le réseau suivant,



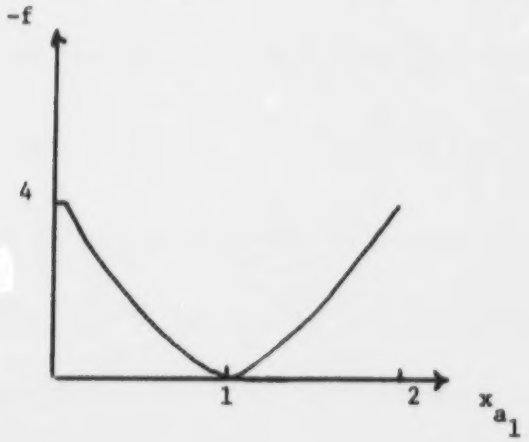
et, $c_{a_1} = 2x_{a_1} + x_{a_2}$

$c_{a_2} = x_{a_1} + 2x_{a_2}$,

un vecteur de coût symétrique, strictement monotone. La solution d'équilibre se trouve à (1,1).

Au point d'équilibre, l'ensemble des solutions réalisables de ce problème sont optimales pour le problème (1.3.2). Par conséquent, f n'est pas différentiable à ce point. Examinons le graphique suivant pour s'en convaincre définitivement.

Graphique 1.3.2 Représentation de la fonction objectif du problème MAX-MIN dans l'exemple 1.3.2.



Même si l'hypothèse de monotonie sur la fonction de coût C n'est pas suffisante pour s'assurer la concavité de f, il n'en reste pas moins que f est concave sur Ω pour une large classe de fonctions de coût dans la réalité.

Proposition 1.3.2:

Supposons que la fonction de coût est affine, c'est-à-dire,

$$C(x) = Ax + b, \text{ où } A \in \mathcal{M}_{M\times M}, \; b \in \mathbb{R}^M,$$

Alors

si la fonction de coût C(x) est monotone sur Ω, la fonction f définie en (1.3.2) est concave sur Ω.

Démonstration:

C(x) est monotone sur Ω

$$\Leftrightarrow$$

$(C(x) - C(y),x-y) \geq 0$ pour tout $x,y \in \Omega$

$$\Leftrightarrow$$

$(C(x) - C(y),x-z) \geq (C(x) - C(y),y-z)$ pour tout $x,y\in\Omega$, $z\in\Omega$ fixé

$$\Leftrightarrow$$

$(C(x) - C(y),x-z) \geq (A(x-y),y-z)$ pour tout $x,y\in\Omega$, $z\in\Omega$ fixé

$$\Leftrightarrow$$

$(C(x),x-z) \geq (C(y),y-z) + (C(y) + A^t(y-z),x-y)$ pour tout $x,y\in\Omega$, $z\in\Omega$ fixé.

La fonction $x \mapsto (C(x),x-z)$ est donc convexe sur Ω pour tout $z\in\Omega$ fixé.

Par conséquent, lorsque C(x) est affine et monotone sur Ω, alors la fonction f est concave sur Ω. □

Nous pouvons aussi réécrire la formulation duale à celle de Smith (1979) sous la forme d'un problème Min-Max comme suit:

$$\min_{x\in\Omega} g(x) \qquad (1.3.3)$$

où

$$g(x) = \max_{y\in\Omega} (C(y), x-y) \qquad (1.3.4)$$

On remarque immédiatement que g est convexe mais non-nécessairement différentiable partout sur Ω. Par contre, le résultat suivant suit immédiatement de la preuve de la proposition 1.3.2.

Proposition 1.3.3:

Supposons que la fonction de coût est affine et strictement monotone sur Ω, alors g est différentiable sur Ω.

Au chapitre 1 (exemple 1.1.3), nous sommes en présence d'une situation où la fonction g n'est pas différentiable partout tandis que la fonction de coût est affine mais non-monotone sur Ω. Par contre, nous n'avons pas pu trouver un exemple de non-différentiabilité de g lorsque la fonction de coût est strictement monotone. Les essais effectués nous ont constamment ramenés à une situation où le vecteur coût est monotone seulement.

Afin d'analyser davantage le comportement de la fonction g et de prendre conscience de la difficulté qui intervient lors du calcul de g à un point $x\in\Omega$ donné, nous allons considérer l'évaluation de g au point x comme un problème d'optimisation.

En effet, nous savons que [Luenberger,p.110,1973]:

"Si y est un minimum local de $(C(y), y-\bar{x})$ sur Ω où $\bar{x}\in\Omega$ est fixé, et C(y) est différentiable sur l'ensemble Ω, alors

$$(\tilde{C}(\bar{y}), y-\bar{y}) \geq 0 \text{ pour tout } y\in\Omega \ . \qquad (1.3.5)$$

où

$\tilde{C}(y) = C(y) + J_C(y)^t(y-\bar{x})$, $J_C(y)$ désignant le Jacobien de C au point y et t l'opérateur transposée."

La relation (1.3.5) correspond alors à la recherche d'un point d'équilibre où l'on considère les coûts marginaux $\tilde{C}(y)$. Dans le cas où C(y) est différentiable 2 fois, le Jacobien de $\tilde{C}$ au point y est symétrique [Luenberger,p.330,1973]. Par conséquent, la relation (1.3.5) est un problème d'équilibre avec des coûts de transport "symétriques".

D'où, l'évaluation de g à un point x donné est tout aussi difficile que de rechercher un flot d'équilibre x^*.

De plus, ces derniers développements mathématiques nous donnent des conditions suffisantes pour la différentiabilité de g. En effet, lorsque le vecteur C(y) est strictement monotone sur Ω, une condition suffisante pour la stricte monotonie de $\tilde{C}(y)$ sur Ω est la monotonie de

$$J_C(y)^t(y-\bar{x}) \quad \text{sur } \Omega, \ \bar{x}\in\Omega \text{ fixé.}$$

On peut vérifier que cette dernière condition est satisfaite lorsque C(x) est affine.

Proposition 1.3.4:

Si le vecteur de coût C(y) est différentiable 2 fois et strictement monotone sur Ω, alors une condition suffisante de la différentiabilité de g est la monotonie de

$$J_C(y)^t(y-\bar{x}) \text{ sur } \Omega, \text{ pour tout } \bar{x}\in\Omega.$$

Nous terminerons cette section en formulant le problème de la recherche d'un flot d'équilibre comme celui de la recherche d'un point de selle. Nous supposerons dans ce qui suit que le vecteur de coût est continu et monotone sur Ω. Par contre, d'autres conditions pourraient être données (voir chapitre 9).

Démontrons la proposition suivante:

Proposition 1.3.5:

Lorsque C(x) est continue et monotone sur Ω, alors $(x^*,y^*)\in\Omega\times\Omega$ est un point de selle de (C(x), x-y) sur $\Omega\times\Omega$ si et seulement si x^* et y^* sont des flots d'équilibre.

Démonstration:

($\Rightarrow$)

Supposons que x^* et y^* sont des flots d'équilibre, cela implique la relation suivante

$$(C(x^*),x^*-y) \leq 0 \leq (C(x),x-y^*) \quad \text{pour tout } x,y\in\Omega,$$

d'après la relation (1.2.2.2) et la proposition (1.3.1). De plus, étant donné que C(x) est monotone sur Ω, on obtient:

$$0 \geq (C(x^*),x^*-y^*) \geq (C(y^*), x^*-y^*) \geq 0$$

ce qui donne

$$(C(x^*),x^*-y^*) = 0$$

$\therefore (x^*,y^*)$ est un point de selle de $(C(x),x-y)$ sur $\Omega\times\Omega$.

($\Leftarrow$)

Supposons que $(x^*,y^*)\in\Omega\times\Omega$ est un point de selle de la fonction $(C(x),x-y)$ sur $\Omega\times\Omega$. Nous obtenons alors

$$\max_{y\in\Omega} (C(x^*),x^*-y) \leq 0 \leq \min_{x\in\Omega} (C(x),x-y^*),$$

c'est-à-dire

$$(C(x^*),y-x^*) \geq 0 \quad \text{pour tout } y\in\Omega$$

et,

$$(C(x),x-y^*) \geq 0 \quad \text{pour tout } x\in\Omega$$

x^* et y^* sont des flots d'équilibre. □

Le corollaire suivant a déjà été démontré par C.S. Fisk & D.E. Boyce (1981).

Corollaire 1.3.5:

Supposons que les fonctions de coût sont strictement monotones, alors x^* est l'unique flot d'équilibre si et seulement si (x^*,x^*) est l'unique point de selle de $(C(x),x-y)$ sur $\Omega\times\Omega$.

Les formulations explicitées dans cette section pour calculer un flot d'équilibre seront utilisées dans les prochains chapitres pour développer des algorithmes de résolution. La prochaine étape consiste à déterminer des conditions d'existence et d'unicité de l'équilibre.

1.4 EXISTENCE ET UNICITE D'UNE SOLUTION D'EQUILIBRE

Considérons en premier lieu l'inégalité variationnelle de Smith (1979). Celui-ci a déjà donné des conditions d'existence et d'unicité de l'équilibre. Elles s'énoncent comme suit:

> Sous l'hypothèse de la continuité de la fonction de coût $C(x)$, il existe une solution d'équilibre. Sous des hypothèses de continuité et de stricte monotonie de $C(x)$, l'unicité de l'équilibre est assurée.

L'hypothèse de stricte monotonie est une condition suffisante mais non-nécessaire pour assurer l'unicité de l'équilibre. Nous verrons dans cette section une autre condition qui remplit le même rôle.

Dans cette perspective, nous allons examiner la formulation duale à celle de Smith (1979). Malheureusement, il n'est pas aussi facile de trouver des conditions d'existence et d'unicité dans ce cas. En effet, l'hypothèse de continuité seulement ne nous assure pas l'existence d'une solution (voir exemple 9.1.3, chapitre 9). Par contre, la continuité et la stricte monotonie de $C(x)$ nous assure l'existence et l'unicité de l'équilibre découlant de la formulation duale de Smith.

Le résultat qui suit a été démontré au chapitre 9 dans le cadre de la programmation convexe.

Proposition 1.4.1

Si la fonction $x \mid\!\rightarrow (C(x), x-y)$ pour tout $y \in \Omega$ fixé est quasi-convexe sur Ω, et $C(x)$ est continue sur Ω, alors il existe une solution à la formulation duale de celle de Smith (1979).

Preuve:

Voir la proposition 9.1.2 et le corollaire 9.1.2 □

Or, la condition précédente entraîne l'équivalence entre les 2 formulations d'équilibre (celle de Smith et celle duale à Smith). En effet, nous avons:

Proposition 1.4.2:

Supposons la quasi-convexité de $x \mid\!\rightarrow (C(x),x-y)$ pour tout $y \in \Omega$ fixé, où $C(x)$ est continue sur Ω, alors on a l'équivalence des 2 relations suivantes:

a) $(C(x^*),x-x^*) \geq 0$ pour tout $x \in \Omega$

b) $(C(x),x-x^*) \geq 0$ pour tout $x \in \Omega$

Démonstration:

(b) $\Rightarrow$ (a)

par continuité de $C(x)$.

(a) $\Rightarrow$ (b)

par définition de quasi-convexité, nous obtenons:

Pour tout $y \in \Omega$, pour tout $x,z \in \Omega$,

$$(C(x),x-y) \leq (C(z),z-y) \Rightarrow (C(z) + J_C(z)^t (z-y)), z-x) \geq 0.$$

Considérons le cas où z=y, nous avons:

pour tout $x,z\in\Omega$, $(C(x),x-z) \leq 0 \Rightarrow (C(z),z-x) \geq 0$.

Par conséquent, en posant $x=x^*$, solution de l'inégalité variationnelle de Smith, il s'en suit:

$(C(z),z-x^*) \geq 0$ pour tout $z\in\Omega$. □

Examinons maintenant les liens qui existent entre la monotonie de C(x) et la quasi-convexité de la fonction

$x \mapsto (C(x),x-y)$ pour tout $y\in\Omega$ fixé.

Pour y arriver, nous allons étudier les 2 exemples suivants.

Exemple 1.4.1

Soit $\Omega = \{(x_1,x_2,x_3) | x_1 + x_2 + x_3 = 2,\ x_1,\ x_2,\ x_3 \geq 0\}$

l'ensemble des flots réalisables, $C((x_1,x_2,x_3)) = (2x_1 + x_2 + x_3,\ x_1 - x_2 - x_3,\ x_1 - x_2 - x_3)$ la fonction de délai, alors le Jacobien de la fonction de coût s'écrit:

$$\begin{bmatrix} 2 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

Posons $x = (x_1,x_2,x_3)$, $\bar{x} = (\bar{x}_1,\bar{x}_2,\bar{x}_3)$ des vecteurs de flot réalisable, alors $(x-\bar{x},\ J_C(x)\ (x-\bar{x})) = -(x_1-\bar{x}_1)^2 < 0$ pour tout $x_1 \neq \bar{x}_1$. Par conséquent, la fonction de coût n'est pas monotone sur le domaine réalisable.

Par ailleurs, il nous faut examiner le comportement de la fonction $x \mapsto (C(x),x-y)$ pour tout $y\in\Omega$ fixé.

On a:

$(C(x),x-y) = (4-x_1)(x_1-y_1)$ et

$(C(z) + J_C(z)^t(z-y),\ z-x) = (z_1-x_1)(4-2x_1+y_1)$.

D'où,

$$(C(z) + J_C(z)(z-y),\ z-x) < 0 \quad \equiv \quad z_1 < x_1 \quad \text{et} \quad (x_1,y_1) \neq (2,0). \qquad (1.4.1)$$

De même,

$$(C(x),\ x-y) \leq (C(z),\ z-y) \quad \equiv \quad (4-x_1)(x_1-y_1) \leq (4-z_1)(z_1-y_1) \qquad (1.4.2)$$

Supposons que la relation (1.4.1) soit vérifiée, examinons alors les cas suivants:

a) $x_1 \geq y_1$

Cela implique que (1.4.2) est non satisfaite.

b) $x < y_1$

Nous obtenons ainsi $0 < (4-x_1) < (4-z_1)$

$$0 > (x_1-y_1) > (z_1-y_1)$$

et la relation (1.4.2) n'est toujours pas satisfaite.

Par conséquent,

$(C(x),\ x-y) \leq (C(z),\ z-y) \Rightarrow (C(z) + J_C(z)^t\ (z-y),\ z-x) \geq 0$ et la quasi-convexité de la fonction $x \mid\!\rightarrow (C(x),\ x-y)$ $y \in \Omega$ fixé est vérifiée.

Exemple 1.4.2

Nous allons reprendre l'exemple 1.3.1 de la section 1.3.

Dans cet exemple, la fonction de coût est monotone sur Ω.

De plus,

$$(C(x),\ x-y) = -(x_{a_1} - y_{a_1})\ (2+\cos x_{a_1})$$

et

$$(C(z) + J_C(z)^t(z-y),\ z-x) = (z_{a_1} - x_{a_1})(-2-\cos z_{a_1} + (z_{a_1} - y_{a_1})\ \sin z_{a_1}).$$

Posons $y_{a_1} = 0$, $x_{a_1} = 2{,}72$, $z_{a_1} = 2{,}74$. Il s'en suit:

$$(C(x),\ x-y) \approx -2.958167657 < -2.957995448 \approx (C(z),\ z-y),$$

$$(C(z) + J_C(z)^t\ (z-y),\ z-x) \approx -0.008536088 < 0.$$

Par conséquent, la fonction $(C(x),\ x-y)$ $y\in\Omega$ fixé n'est pas quasi-convexe sur Ω.

Les exemples (1.4.1) et (1.4.2) démontrent bien l'indépendance entre la monotonie de la fonction de coût et la quasi-convexité de la fonction $x \mapsto (C(x),\ x-y)$, y fixé.

A partir des résultats de cette section, nous obtenons une nouvelle condition d'existence et d'unicité de l'équilibre.

Proposition 1.4.3:

En supposant la stricte quasi-convexité de la fonction $(C(x),x-y)$, $y\in\Omega$ fixé et la continuité de la fonction de coût $C(x)$, nous sommes assurés de l'existence et de l'unicité de l'équilibre obéissant au premier principe de Wardrop.

Démonstration:

Le résultat est obtenu grâce aux propositions (1.4.1) et (1.4.2). □

Pour terminer cette section, nous aborderons le deuxième principe de Wardrop (Optimum collectif).

Comme nous le savons, le problème de la recherche d'un flot d'optimum collectif s'écrit:

$$\text{Min } (C(x),\ x) \qquad (1.4.3)$$

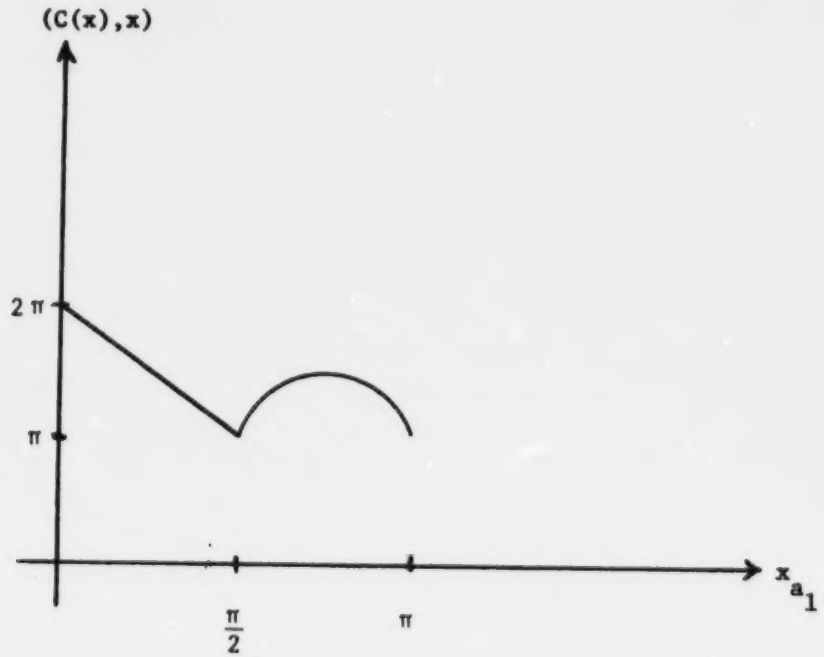
sujet à

$$x\in\Omega.$$

Même si la fonction objectif de (1.4.3) est strictement convexe sur Ω dans certains cas particuliers (voir Fisk & Nguyen, 1980b), il n'en est pas moins clair que la stricte convexité de $(C(x),x)$ sur Ω n'est pas toujours satisfaite de façon générale.

Pour s'en convaincre, considérons de nouveau l'exemple 1.3.1. Nous savons que la fonction de coût dans cet exemple est monotone sur Ω. Dans le graphique 1.4.1, nous avons tracé la courbe de la fonction (C(x),x) sur l'ensemble réalisable Ω. Nous voyons bien que (C(x), x) n'est pas convexe sur Ω.

Dans les prochains chapitres, nous verrons différentes approches de résolution des problèmes d'équilibre.



Graphique 1.4.1 - Représentation de la fonction objectif du problème d'optimum collectif dans l'exemple 1.3.1.

CHAPITRE 2

RECHERCHE D'UN POINT D'EQUILIBRE LORSQUE LA DEMANDE EST FIXE

2.1 PROBLEME DE BECKMANN: COUTS SYMETRIQUES.

La recherche d'un flot d'équilibre (Optimum-égoïste) sur un réseau routier a été examinée par beaucoup de chercheurs. L'équivalence entre l'affectation du trafic d'équilibre sur un réseau routier et la minimisation d'une certaine fonction convexe des trafics a été originellement démontrée par Beckmann et al (1956). En supposant que la fonction de coût sur chaque arc dépend uniquement du flot sur cet arc et que cette fonction est continue et croissante, le modèle mathématique a été repris et analysé par plusieurs auteurs (Jewell, 1967; Dafermos et Sparrow, 1969; Potts & Oliver, 1972; Leventhal et al., 1973; Nguyen, 1974).

Bien que le modèle "standard" soit valable dans certaines applications, il est impossible de tenir compte des intéractions entre les arcs du réseau. Un modèle étendu a été proposé où les fonctions de coût sur chaque arc dépendent possiblement du flot sur chaque arc du réseau (Dafermos, 1971). Lorsque la fonction de coût $C(x)$ est le gradient d'une fonction convexe $f(x)$, l'existence d'un problème d'optimisation équivalent est démontrée (Dafermos, 1972). Cela est dû au résultat suivant: $C(x)$ est le gradient d'une fonction $f(x)$ si et seulement si le Jacobien de la fonction de coût, dénote $J_C(x)$ est symétrique.

Cette extension est intéressante étant donné que nous pouvons modéliser des réseaux multimodaux après des transformations appropriées.

Plusieurs approches de résolution ont été envisagées pour résoudre le problème de Beckmann (1956). Des méthodes de directions réalisables où la direction de descente est obtenue soit par approximation linéaire de la fonction objective de Beckmann ou par une partition des variables en variables de base et variables hors-base, des algorithmes basés sur la génération (a priori/progressive) des chemins entre les zones et enfin, une méthode exploitant la structure d'arbre des bases ont été développés par Nguyen (1974), Nguyen (1976) et Leblanc et al. (1975).

Plusieurs techniques d'accélération de convergence ont aussi été examinées par la suite. Marc Shaigetz (1978) tente d'améliorer le temps d'exécution et la vitesse de convergence d'un algorithme développé par Frank et Wolfe (1956) basé sur l'approximation linéaire d'une fonction économique convexe, sous des contraintes linéaires. Il considère l'algorithme de Frank et Wolfe dans le contexte d'une stratégie de restriction et de linéarisation interne du domaine réalisable (C.A. Holloway, 1974) et s'inspire de Wolfe (1970) pour obtenir une nouvelle direction de descente ("Away step"). Une extension de l'algorithme de Frank et Wolfe basée sur plusieurs directions de descente est aussi proposée. De plus, Florian (1977) suggère une variante de l'adaptation de l'approche de Wolfe (1970) au problème d'affectation de Beckmann (1956). Florian & Guélat (1983) considère aussi la méthode PARTAN (D.G. Luenberger, 73) pour améliorer la convergence de l'algorithme de Frank & Wolfe.

Ce modèle "étendu" n'est pas encore entièrement satisfaisant. En effet, l'hypothèse de symétrie du Jacobien du vecteur de coût est très restrictive et c'est invraisemblable qu'elle tienne dans la majorité des applications. Nous allons relaxer l'hypothèse de symétrie tout en conservant l'hypothèse sur la monotonie des fonctions de coûts. Cette dernière est relativement raisonnable; et elle est satisfaite en générale lorsque le coût sur un arc donné dépend principalement du flot sur cet arc.

Malheureusement, il n'existe plus de formulation d'optimisation de Beckmann lorsque les fonctions de coût sont asymétriques (Jacobiens non symétriques). Par contre, nous avons développé dans les chapitres précédents une nouvelle formulation d'optimisation convexe pour résoudre ce problème d'équilibre général. Nous utiliserons celle-ci pour développer de nouvelles approches de calcul.

2.2 PROBLEME GENERAL: COUTS ASYMETRIQUES

Nous développons dans cette section une nouvelle méthode d'accumulation de contraintes pour résoudre le problème d'équilibre général lorsque les fonctions de coût sont strictement monotones sur l'ensemble des flots réalisables.

2.2.1 Introduction aux méthodes de coupe

Nous aborderons la résolution du problème de la recherche d'un flot d'équilibre sur un réseau routier à partir de la formulation MIN-MAX suivante:

$$\text{Min } g(x) \qquad (2.2.1.1)$$

sujet à

$$x \in \Omega$$

où

$$g(x) = \max_{y \in \Omega} (C(y), x-y). \qquad (2.2.1.2)$$

Nous utiliserons le même cheminement que Gribik (1978) pour se donner des algorithmes facilement implantables et de plus en plus performants.

Puisque g est une fonction convexe, mais non-nécessairement différentiable partout sur Ω, et que Ω est un ensemble convexe et compact, nous pouvons utiliser une méthode de plan de coupe pour résoudre le problème d'équilibre (2.2.1.1 - 2.2.1.2). Pour se faire, nous ramenons la fonction g(x) sous forme de contraintes dans (2.2.1.1).

La méthode de plan de coupe convexe introduite par Kelley (1960) et Cheney et Goldstein (1959), adaptée à notre problème s'énonce ainsi:

Algorithme 2.2.1.1:

Etape 0. Déterminer un point réalisable $x' \in \Omega$, poser $k = 1$.

Etape 1. Trouver y^k une solution optimale (globale) du problème (2.2.1.2) lorsque $x = x^k$.

Etape 2. Trouver x^{k+1} une solution optimale du problème linéaire

$$\min_{x \in \Omega} \max_{1 \le i \le k} (C(y^i), x-y^i). \qquad (2.2.1.3)$$

Poser $k \leftarrow k+1$.

Retourner à l'étape 1.

La forme générale de l'algorithme de plan de coupe pour un programme convexe est donnée dans Zangwill (1969), Luenberger (1973). La procédure consiste à calculer la coupe la plus "profonde" permettant d'éliminer la solution optimale courante x^k et à raffiner de plus en plus l'approximation obtenue du problème à résoudre.

Kelly (1960) démontre l'existence d'une sous-suite K de $\{x^k\}$ qui converge vers une solution d'équilibre x*. De plus, sous l'hypothèse de stricte monotonie de la fonction de coût, la sous-suite $\{y^k\}$ converge aussi vers x*.

L'implantation de cet algorithme est liée à la résolution du problème (2.2.1.2) à l'étape 1. En effet, le calcul d'un sous-gradient de g au point x^k (ou encore, la recherche du point y^k) et l'évaluation de la fonction g au point x^k ne sont pas réalisables sauf pour des cas très simples. C'est pour cette raison que nous allons examiner des plans de coupe approximatifs.

Un premier critère d'approximation est donné par Parikh (1976); chaque hyperplan de coupe dénoté $H(x;x^k)$ doit satisfaire les 2 conditions suivantes à savoir:

a) $g(x^k) - H(x^k;x^k) < \varepsilon, \quad \varepsilon > 0$

b) $g(x) \geq H(x;x^k)$ pour tout $x \in \Omega$, $k = 1,2,\ldots$

Par conséquent, on peut considérer l'hyperplan $H(x;x^k) = (C(y_\varepsilon^k), x-y_\varepsilon^k)$, où y_ε^k est une solution ε-optimale de (2.2.1.2) avec $x=x^k$.

Dans ce nouvel algorithme, la constante $\varepsilon>0$ a été fixée une fois pour toutes au début de la procédure. Gustafson et Kortanek (1973) considèrent plutôt la suite $\{\varepsilon_k\}_{k=1}^{\infty}$ de nombres non-négatifs telle que $\lim_{k\to\infty} \varepsilon_k \neq 0$. En s'assurant que pour tout k, $(C(y_{\varepsilon_k}^k), x^k - y_{\varepsilon_k}^k) > 0$, ils prouvent la convergence. Cette légère modification semble opportune car il n'est pas justifié de calculer $y_{\varepsilon_k}^k$ avec ε_k très petit lorsque k est petit. Par contre, la précision devra être de plus en plus grande à mesure que k croît.

Puisque ces approximations nous permettent de résoudre une classe restreinte de problèmes seulement, nous allons considérer d'autres approches plus simples pour déterminer des plans de coupe.

Pour se faire, nous allons envisager le problème Min-Max (2.2.1.1-2.2.1.2) comme un problème de programmation linéaire ayant un nombre infini de contraintes c'est-à-dire,

$$\text{Min } x_o$$

sujet

$$(C(y^i), x-y^i) \leq x_o, \quad i = 1,2,\ldots,\infty \qquad (2.2.1.4)$$

$x \in \Omega$.

Chacun des algorithmes précédents utilise une approche de restriction sur le dual de (2.2.1.4) qui consiste à permettre à un nombre fini de variables à être non-nulles, toutes les autres variables duales sont mises à zéro. Cette procédure nous a amenés à résoudre le problème primal restreint à $y^1,\ldots,y^k$ à l'itération k.

Considérons maintenant les coefficients de coût relatif du dual de (2.2.1.3) à l'optimum. Ils sont de la forme

$$(C(y),\ y-x^{k+1}) - g(x^{k+1}),\ y \in \Omega.$$

Si $y \in \{y^1,y^2,\ldots,y^k\}$ alors le coefficient de coût relatif est non-négatif. Par contre, si l'équilibre n'est pas atteint, il existe un $y \in \Omega$ tel que $(C(y),\ y-x^{k+1}) - g(x^{k+1}) < 0$. Pour déterminer un tel y, nous avons jusqu'à maintenant tenter de réduire au mieux le terme $(C(y),y-x^{k+1})$. Or, il n'est pas nécessaire d'agir ainsi car la fonction g est strictement positive lorsque nous ne sommes pas à l'équilibre.

Par conséquent, nous pouvons choisir comme valeur de y le point x^{k+1} et nous obtenons alors un hyperplan de la forme $(C(x^{k+1}),x-x^{k+1})$ à l'itération k+1. L'algorithme obtenu a été établi initialement par Zuhovickii, Polyak et Primak (1969) pour la résolution de problèmes de jeux à n-personnes. Malheureusement, cette méthode simple possède un taux lent de convergence (Nguyen et Dupuis, 1981).

Dans la prochaine section, en se basant sur les remarques précédentes, nous allons construire une nouvelle coupe de manière à allier la vitesse de convergence et la simplicité d'implantation.

2.2.2 Méthode d'accumulation des contraintes

L'algorithme que nous décrirons ici a été étudié en détail dans Nguyen et Dupuis (1981). Nous ne ferons ici qu'énoncer l'algorithme et donner une preuve de convergence, l'implantation sera examinée en détail au chapitre suivant.

Nous savons que le problème MIN-MAX précédent (2.2.1.1)-(2.2.1.2) peut s'écrire comme suit:

$$\text{Max } x_o$$

$$(C(x),\ x-u) \geq x_o \quad \text{pour tout } x\in\Omega \qquad (2.2.2.1)$$
$$u\in\Omega$$

Or, le problème (2.2.2.1) est linéaire en x_o et u; mais le nombre de contraintes est infini. De plus, l'ensemble des points réalisables ne peut être décrit explicitement.

La stratégie utilisée est la suivante; supposons connus les points $x^1, x^2, \ldots, x^k$ dans Ω, on peut alors approximer (2.2.2.1) par le problème linéaire suivant possédant un nombre fini de contraintes:

$$\text{Max } x_o$$

$$(C(x^i),\ x^i-u) \geq x_o, \quad i = 1,2,\ldots,k \qquad (2.2.2.2)$$
$$u\in\Omega$$

susceptible d'être résolu efficacement par une adaptation appropriée de la méthode du simplexe. Lorsqu'une solution optimale de (2.2.2.2) est réalisable pour le problème (2.2.2.1), il est clair qu'elle est aussi optimale pour (2.2.2.1). Dans le cas contraire, nous pouvons raffiner l'approximation de (2.2.2.1) en ajoutant à (2.2.2.2) une nouvelle contrainte éliminant la solution optimale courante du problème linéaire et ainsi de suite.

La stratégie que l'on vient de décrire en termes généraux est le concept de base des méthodes dites méthodes d'accumulation de contraintes ou encore méthodes des hyperplans de séparation dont les premières ont été introduites par Kelley (1960) et Cheney et Goldstein (1959).

En adoptant le formalisme développé par Zangwill (1969), nous allons maintenant décrire un algorithme général d'accumulation de contraintes. Nous élaborerons ensuite un algorithme spécifique pour résoudre (2.2.2.1) et nous montrerons sa convergence.

Considérons le problème général

$$\underset{u\in S\subset R^N}{\text{Max } F(u)} \qquad (2.2.2.3)$$

où F(u) est une fonction à valeur réelle continue sur le compact non-vide S. La méthode générale d'accumulation des contraintes fonctionne comme suit: initialement, on se donne un compact $Z' \supseteq S$; à l'étape k, supposons donné Z^k, on calcule alors $u^k \in \Gamma Z^k$ où

$$\Gamma Z^k = \{u^* \in Z^k \mid F(u^*) = \text{Max } F(u),\ u \in Z^k\};$$

si $u^k \in S$, c'est terminé; sinon, il s'agit de déterminer $v^k \in \Delta u^k$, où Δ est une multi-application à préciser et un demi-espace fermé $H(v^k)$, tel que:

$$H(v^k) \supseteq S \quad \text{et} \quad u^k \notin H(v^k). \tag{2.2.2.4}$$

Nous posons alors

$$Z^{k+1} = Z^k \cap H(v^k) \tag{2.2.2.5}$$

de sorte que

$$Z^k \supset Z^{k+1} \supseteq S, \tag{2.2.2.6}$$

et,

$$F(u^k) \geq F(u^{k+1}) \geq \max_{u \in S} F(u) \tag{2.2.2.7}$$

La relation (2.2.2.7) nous fait remarquer que si $u^k \in S$, cela entraîne du même coup que u^k est solution optimale du problème (2.2.2.3).

Pour vérifier si $u^k \in S$ ou non, nous devons choisir un critère d'arrêt approprié tel que si u^k vérifie ce critère, on sera en mesure d'identifier une solution optimale du problème original. Si nous construisons une suite infinie de points $\{u^k\}_1^\infty$, il faut s'assurer que la limite $\bar{u}$ de toute sous-suite convergente (une telle sous-suite existe toujours puisque tous les u^k sont dans un compact) satisfait le critère d'arrêt choisi.

Lorsque H(v) est définie par une contrainte linéaire de la forme:

$$H(v) = \{u \mid (a(v), u) + b(v) \geq 0\} \tag{2.2.2.8}$$

où $a(v) \in R^N$, $b(v) \in R^1$ sont des fonctions continues, une condition suffisante de convergence établie par Zangwill (1969) s'énonce comme suit:

Théorème 2.2.2.1:

Si aucun des u^k ne vérifie le critère d'arrêt, si tous les points de la suite $\{v^k\}_1^\infty$ sont dans un compact, et la multi-application Δu est fermée en tout point u^k, alors la limite de toute sous-suite convergente satisfait le critère d'arrêt.

Démonstration: Comme v^k et u^k sont dans des compacts, il doit exister un ensemble L, tel que:

$$u^k \to \bar{u}, \qquad k \in L$$
$$v^k \to \bar{v}, \qquad k \in L .$$

Supposons que $\bar{u}$ ne vérifie pas le critère d'arrêt, on a

$$\bar{v} \in \Delta\bar{u},$$

puisque Δ est fermée, et nous avons

$$\bar{u} \notin H(\bar{v}) \tag{2.2.2.9}$$

D'autre part, d'après (2.2.2.5) - (2.2.2.6),

$$u^\ell \in H(v^k) \qquad \text{pour tout } 1 \geq k+1,$$

d'où,

$$(a(v^k),\bar{u}) + b(v^k) \geq 0, \tag{2.2.2.10}$$

et, en faisant tendre $v^k \to \bar{v}$, on obtient

$$(a(\bar{v}),\bar{u}) + b(\bar{v}) \geq 0,$$

ce qui contredit (2.2.2.9), donc $\bar{u}$ doit vérifier le critère d'arrêt. □

Considérons maintenant la procédure de calcul suivante pour résoudre (2.2.2.1):

Algorithme 2.2.2.1:

1. Initialisation

Nous déterminons un flot réalisable $v' \in \Omega$ et posons

$$Z' = \{(x_o,u) \mid (C(v'), v'-u) \geq x_o,\ u \in \Omega\},\ k \leftarrow 1.$$

2. Résolution du sous-problème linéaire

Déterminer une solution optimale (x_o^k, u^k) de

$$\begin{array}{l} \text{Max } x_o \\ (x_o,u) \in Z^k \end{array} \qquad (2.2.2.11)$$

3. Critère d'arrêt

Si $x_o^k \leq 0$, c'est terminé.

4. Calcul de l'ensemble Z^{k+1} .

Il s'agit de calculer

$$v^{k+1} = v^k + \theta^k\ (u^k - v^k) \qquad (2.2.2.12)$$

où

$$e(v^{k+1}) = \underset{\theta \in [0,1]}{\text{Min}}\ e(v^k + \theta(u^k - v^k)) \qquad (2.2.2.13)$$

et

$$e(v) = \{(C(v),\ v^k - u^k)\}^2. \qquad (2.2.2.14)$$

Puis, nous formons

$$Z^{k+1} = Z^k \cap H(v^{k+1})$$

où

$$H(v^{k+1}) = \{(x_o,u) \mid (C(v^{k+1}),\ v^{k+1} - u) \geq x_o\} \qquad (2.2.2.15)$$

et retournons à l'étape 2 avec $k \leftarrow k+1$.

Nous allons d'abord montrer que l'algorithme ci-dessus converge à l'aide du théorème de convergence énoncé plus tôt, et montrer ensuite que l'on résoud le problème (2.2.2.1).

Supposons que l'algorithme ci-dessus construit une suite infinie de points $\{x_o^k, u^k, v^k\}_1^\infty$. Remarquons immédiatement que tous les points x_o^k, u^k, v^k sont dans des compacts, notamment

$$x_o^k \in [0, x_o^1]$$

$$u^k, v^k \in \Omega.$$

Nous devons alors montrer 1) que la multi-application $\Delta(u^k, v^k) = \{v^{k+1}\}$ définie par (2.2.2.12 - 2.2.2.14) est fermée, et 2) que le demi-espace $H(v^{k+1})$ défini par la relation (2.2.2.15) est bien de la forme (2.2.2.8) et satisfait aux conditions (2.2.2.4). La convergence de l'algorithme est alors assurée par le théorème de convergence énoncé plus tôt.

1) <u>$\Delta(u^k, v^k)$ est une multi-application fermée</u>:

En effet, il doit exister un K tel que

$$u^k \to u^\infty \qquad k \in K$$
$$v^k \to v^\infty \qquad k \in K$$

et $\quad v^{k+1} \in \Delta(u^k, v^k)$,

on doit montrer que:

$$v^{\infty+1} \in \Delta(u^\infty, v^\infty). \tag{2.2.2.16}$$

Tous les $\theta^k \in [0,1]$ impliquent qu'il existe un $K_1 \subset K$ tel que:

$$\theta^k \to \theta^\infty \qquad k \in K_1,$$

et par conséquent (2.2.2.12) donne

$$v^{\infty+1} = v^\infty + \theta^\infty (u^\infty - v^\infty). \tag{2.2.2.17}$$

D'après la relation (2.2.2.13), on a

$$e(v^{k+1}) \leq e(v^k + \theta(u^k - v^k)), \text{ pour tout } \theta \in [0,1]$$

e(v) est continue puisque C(v) l'est, en passant à la limite, on obtient:

$$e(v^{\infty+1}) \leq e(v^\infty + \theta(u^\infty - v^\infty)). \tag{2.2.2.18}$$

Soit

$$\bar{v} \in \Delta(u^{\infty}, v^{\infty}), \tag{2.2.2.19}$$

la relation (2.2.2.18) étant valide pour tout $\theta \in [0,1]$, on obtient

$$e(v^{\infty+1}) \leq e(\bar{v}). \tag{2.2.2.20}$$

Les relations (2.2.2.17), (2.2.2.19) et (2.2.2.20) impliquent donc (2.2.2.16).

2) Validité de H(v):

C est supposé strictement monotone sur Ω. H(v) est bien de la forme (2.2.2.8), puisque C(v) et (C(v),v) sont continues. D'autre part, quel que soit k, on a toujours

$$Z^k \supseteq S$$

où $S = \{(x_o,u) \in [-M,0] \times \Omega \,|\, (C(x), x-u) \geq x_o \text{ pour tout } x \in \Omega\}$,

$$-M = \underset{u\in\Omega}{\text{Min}}\ \underset{x\in\Omega}{\text{Min}}\ (C(x), x-u).$$

Il reste à montrer que si $x_o^k \geq 0$,

$$(x_o^k, u^k) \notin H(v^{k+1}).$$

En effet, en posant

$$r(\theta) = (C(v(\theta)), v^k - u^k) \tag{2.2.2.21}$$

où

$$v(\theta) = v^k + \theta(u^k - v^k),\ \theta \in [0,1], \tag{2.2.2.22}$$

La fonction $r(\theta)$ est continue et strictement monotone décroissante puisque

$$(C(v(\theta_1)) - C(v(\theta_2)), v(\theta_1) - v(\theta_2)) > 0 \qquad \theta_1 \neq \theta_2 \in [0,1]$$

entraîne l'inégalité:

$$(r(\theta_1) - r(\theta_2),\ \theta_1 - \theta_2) < 0 \qquad \theta_1 \neq \theta_2 \in [0,1].$$

Par définition,

$$[r(\theta^k)]^2 = \text{Min}\{[r(\theta)]^2 \,|\, \theta \in [0,1]\},$$

sachant que

$$r(0) = (C(v^k),\ v^k - u^k) = x_o^k > 0$$

On déduit immédiatement que $\theta^k > 0$ et plus précisément

$$r(\theta^k) = \begin{cases} r(1) & \text{si } r(1) > 0 \\ 0 & \text{autrement} \end{cases}$$

On peut donc écrire

$$(1-\theta^k)r(\theta^k) = 0$$

ou de façon équivalente

$$(C(v^{k+1}),\ v^{k+1}-u^k) = 0$$

ce qui montre

$$(x_o^k, u^k) \notin H(v^{k+1}).$$

Nous avons montré que l'algorithme précédent converge en accord avec le critère d'arrêt choisi lorsque C est supposé strictement monotone.

Si cette hypothèse de stricte monotonie n'est pas vérifiée, nous pouvons considérer l'ajustement suivant:

"A l'étape 4 de l'algorithme 2.2.2.1, le calcul de v^{k+1} est fait autrement c'est-à-dire, les relations (2.2.2.13) et (2.2.2.14) s'écrivent respectivement:

$$e^1(v^{k+1}) = \operatorname{Min}_{\theta\in[0,1]} e^1\,(v^k+\theta(u^k-v^k)) \qquad (2.2.2.23)$$

et,

$$e^1(v) = (C(v),\ v-u^k)". \qquad (2.2.2.24)$$

Soit $\Delta^1(u^k,v^k)$ la multi-application associée à (2.2.2.23) et (2.2.2.24), celle-ci est fermée à la lumière de la preuve de convergence de l'algorithme 2.2.2.1 (partie 1)).

De plus,

$$(C(v^{k+1}), v^{k+1}-u^k) \leq 0 < x_o^k$$

à partir de (2.2.2.23) - (2.2.2.24).

Par conséquent,

$$(x_o^k, u^k) \notin H(v^{k+1}).$$

La convergence de l'algorithme est encore démontrée. Il reste à vérifier que l'on détermine effectivement une solution optimale à (2.2.2.1).

Théorème 2.2.2.2:

En supposant la stricte monotonie de C sur Ω, peu importe laquelle des 2 options pour calculer v^{k+1} à l'étape 4 de l'algorithme a été choisie, si l'algorithme termine après k itérations, v^k est une solution optimale de (2.2.2.1), sinon il existe une sous-suite $\{v^k\}_{K_1}$ convergeant vers cette solution optimale.

Démonstration:

Soit $\bar{u}$ la solution optimale de (2.2.2.1). L'optimalité de (x_o^k, u^k) par rapport au sous-problème à l'itération k implique:

$$(C(v^i),\ v^i - \bar{u}) = \min_{i \le L \le k} (C(v^L),\ v^L - \bar{u}) \le x_o^k. \qquad (2.2.2.25)$$

D'autre-part, la stricte monotonie de C permet d'écrire

$$(C(v),\ v - \bar{u}) > 0 \qquad \text{pour tout } v \neq \bar{u}$$

de sorte que, si l'algorithme termine après l'itération k, on a

$$x_o^k = 0 \Rightarrow v^i = \bar{u}\ .$$

Puisque

$$x_o^i \le (C(v^i), v^i - u^i) = (C(\bar{u}), \bar{u} - u^i) \le 0$$

à l'itération i et que $x_o^L > 0$ pour tout $L \le k-1$, on a par conséquent

$$k = i \quad \text{et} \quad v^k = v^i = \bar{u}.$$

Considérons maintenant le cas où l'algorithme construit une suite infinie de points $\{x_o^k, v^k\}$, tous dans un compact, et que $x_o^k > 0$. D'après (2.2.2.25), il existe une sous-suite K tel que

$$(C(v^k),\ v^k-\bar{u}) \leq x_o^k,\ k \in K.$$

Sachant que la limite de toute sous-suite convergente $\{x_o^k\}_{K_1}$, $K_1 \subset K$, est nulle, la sous-suite $\{v^k\}_{K_1}$ converge donc vers $\bar{u}$. □

Malheureusement, en l'absence de stricte monotonie, même si l'algorithme converge en choisissant la fonction e^1 pour calculer v^{k+1} à l'étape 4, cela n'entraîne pas nécessairement la convergence de la suite $\{v^k\}$ vers une solution optimale de (2.2.2.1). Il faut s'assurer que la condition suivante soit satisfaite:

$$(C(v),\ v-\bar{u}) > 0 \quad \text{pour tout } v \neq \bar{u} \tag{2.2.2.26}$$

où $\bar{u}$ est une solution optimale quelconque de (2.2.2.1).

Néanmoins, nous pouvons passer outre à la condition (2.2.2.26) en se rappelant que notre objectif n'est pas tellement de résoudre (2.2.2.1) mais plutôt de déterminer une solution à l'inégalité variationnelle de Smith (1979) par l'intermédiaire de (2.2.2.1).

Nous supposons qu'il existe au moins une solution à la formulation duale de Smith (voir proposition 1.3.1) c'est-à-dire,

$$\text{Max}_{x\in\Omega}\ g(x) \equiv 0.$$

Nous obtenons alors l'algorithme suivant.

Algorithme 2.2.2.2

- Les étapes 1 et 2 de l'algorithme 2.2.2.1 et de celui-ci sont identiques.
- Etape 3: Evaluation de $G(u^k)$

  Déterminer une solution w^k de

  $$\text{Min}_{w\in\Omega}\ (C(u^k,w)$$

  poser $G(u^k) = (C(u^k),\ u^k-w^k)$.

. Etape 4: Critère d'arrêt

Si $G(u^k) = 0$ alors terminé.

. Etape 5: Calcul de l'ensemble Z^{k+1}.

Il s'agit de calculer

$$v^{k+1} = u^k + \theta^k(w^k - u^k)$$

où

$$e^1(v^{k+1}) = \underset{\theta\in[0,1]}{\text{Min}} \; e^1(u^k + \theta(w^k - u^k)),$$

e^1 étant définie en (2.2.2.24).

Puis, nous formons

$$Z^{k+1} = Z^k \cap H(v^{k+1})$$

où H(v) est défini en (2.2.2.15).

Retourner à l'étape 2 avec $k \leftarrow k+1$.

Remarquons tout de suite que si le critère d'arrêt est vérifié, alors u^k est une solution optimale de l'inégalité variationnelle de Smith (1979). Puisque toutes les solutions de (2.2.2.1) sont telles que G est minimale c'est-à-dire, à valeur nulle, alors ce critère est justifié.

Par conséquent, il nous reste à vérifier la condition suffisante de convergence de Zangwill (1969).

D'abord, pour démontrer que la multi-application permettant de calculer v^{k+1} est fermée, il s'agit de cerner un ensemble K tel que

$$u^k \to u^\infty, \; k \in K$$
$$v^k \to v^\infty, \; k \in K$$
$$w^k \to w^\infty, \; k \in K,$$

et $$v^{k+1} \in \Delta(u^k, v^k, w^k).$$

On montre que $v^{\infty+1} \in \Delta(u^\infty, v^\infty, w^\infty)$ comme précédemment. Puis, pour se rendre compte que $(x_o^k, u^k) \notin H(v^{k+1})$ si $G(u^k) \neq 0$, cela suit du fait que

$$\frac{d[e^l(u^k+\theta(w^k-u^k))]}{d\theta}\Big/_{\theta=0} = (C(u^k),\ w^k-u^k) < 0,$$

et,

$$e^l(u^k) = 0.$$

Comparativement à l'algorithme (2.2.2.1), l'algorithme (2.2.2.2) exige en plus le calcul non négligeable de w^k mais, n'impose pas l'hypothèse de monotonie.

Remarquons enfin que l'on peut éliminer à chaque étape de ces algorithmes les contraintes non-actives si une diminution suffisante de l'objectif à minimiser est obtenue [Eaves et Zangwill, 1971]. Nous reviendrons sur ce point lors de l'implantation de l'algorithme (2.2.2.1) au chapitre suivant.

Nous verrons ultérieurement que l'algorithme (2.2.2.1) est relié étroitement à l'algorithme de Frank & Wolfe dans le cas "symétrique". Mais auparavant, nous discuterons quelque peu du taux de convergence de la méthode d'accumulation des contraintes.

2.2.3 Taux de convergence de la méthode d'accumulation des contraintes.

Comme dans la plupart des algorithmes basés sur une approche de programmation linéaire, l'étude du taux de convergence des algorithmes de plan de coupe n'est pas satisfaisante.

D'abord, Zuhovickii, Polyak et Primak (1969) nous donnent les estimés suivants concernant leur méthode:

$$\min_{i=1,\ldots,k} (C(x^i),\ x^i-x^{k+1}) = 0(1/k),$$

et,

$$\|x^i-x^*\| = 0(1/k)$$

à la condition que l'ensemble réalisable soit fortement convexe ce qui n'est pas le cas ici.

Levitin et Polyak (1966), Topkis (1969,1970), Elzinga et al. (1975) et Gribik (1978) ont donné des taux de convergence arithmétique pour une classe générale d'algorithmes de plan de coupe englobant ceux de Kelley (1960), Cheney et Goldstein (1959) et une généralisation de la méthode utilisée par Zoutendijk

(1960) et Veinott (1967). Mais leurs hypothèses sont beaucoup trop restrictives.

Nous allons plutôt nous attarder au taux de convergence de la méthode d'accumulation des contraintes dans le cas "symétrique" en se basant sur l'objectif du problème de Beckmann (1956).

Posons $C(x) = \nabla F(x)$ où F est strictement convexe sur Ω.

La proposition suivante est essentielle aux développements subséquents.

Proposition 2.2.3.1:

Considérons la méthode d'accumulation des contraintes (algorithme 2.2.2.1) à l'étape k; soit x^k le point courant, y^k la solution optimale du problème linéaire à l'étape 1, alors

$\bar{s}_k = (y^k - x^k)$ est une direction de descente de F au point x^k.

De plus,

$$\theta^k = \arg \min_{\theta \in [0,1]} \{(C(x^k + \theta \bar{s}_k),\ x^k - y^k)\}$$

est le pas optimal c'est-à-dire,

$$\theta^k = \arg \min_{\theta \in [0,1]} F(x^k + \theta \bar{s}_k) .$$

Démonstration:

Cela suit du fait que $(\nabla f(x^k), \bar{s}_k) = (Cx^k, y^k - x^k) < 0$. □

Nous voulons en premier lieu analyser le comportement de la suite $\{F(x^k)\}$. Pour se faire, nous nous sommes basés sur un résultat de Frank & Wolfe (1956) qui s'énonce comme suit:

Proposition 2.2.3.2:

Soit $z_{k=1,2,...}$ la suite de points générés par l'algorithme de Frank & Wolfe pour résoudre le problème

$$\min_{z\in\Omega} F(z), \qquad (2.2.3.1)$$

soit $x^*\in\Omega$ la solution de (2.2.3.1), alors

$$F(z_k)-F(x^*) < \alpha/k,$$

où α est une constante positive et k est suffisamment grand.

Il nous a permis de trouver un résultat connexe approprié pour la méthode d'accumulation des contraintes dû à Karmanov (Corollaire 1,p.170, 1977). Karmanov démontre la relation

$$F(x^k)-F(x^*) < \bar{\alpha}/k, \quad k = 1,2,...$$

où $\bar{\alpha}$ est une constante positive, à la condition que

$$(C(x^k),x^k-y^k) \geq \sigma_k(C(x^k), x^k-x^*), \quad 0 < \sigma < \sigma_k \text{ pour tout } k.$$

Cette condition s'est avérée vraie en pratique; par exemple, un essai sur le réseau de Hull (voir chapitre 3) a été effectué; à la 285e itération, la diminution de la fonction objectif F était visible après 5 décimales après le point seulement; par contre, la norme $\|x^k-y^k\|_{k=285}$ était supérieure à 10^6.

De plus, sachant que l'optimum du problème linéaire n'est pas unique, si à l'étape k, x^k est le flot d'équilibre, y^k sera généralement différent de x^k. Ces réflexions nous amènent à croire que la condition précédente est vérifiée dans bien des cas.

Canon et Cullum (1968) ont donné une borne supérieure sur le taux de convergence de l'algorithme de Frank et Wolfe. En supposant F convexe, 2 fois continuement différentiables et le Jacobien de la fonction de coût au point x^* défini positif, et en admettant que: 1) $x^*\in\partial\Omega$, la frontière de Ω, c'est-à-dire, x^* appartient à une arête quelconque de Ω, 2) $z \in$ ri Ω, l'intérieur relatif de Ω, pour un nombre infini d'indices $k \in \{1,2,...\}$ alors,

pour tout $\alpha,\varepsilon > 0$, $F(z_k)-F(x^*) > \alpha/k^{1+\varepsilon}$ pour un nombre infini d'indices k.

Les conditions 1) et 2) garantissent que l'algorithme ne termine pas en un nombre fini d'étapes. Dans la majorité des cas, la condition 1) est vérifiée tandis que 2) l'est dans la situation suivante: $z_1 \in$ ri Ω est choisi de sorte que $F(z_1) < F(w^i)$, $i = 1,\ldots,I$ où w^i est un point extrémal de Ω.

Considérons maintenant la méthode d'accumulation des contraintes. Nous allons supposer que les conditions précédentes sont vérifiées pour la suite $\{x^k\}$, tandis que la suite $\{y^k\}$ satisfait la relation $y^k \notin$ ri Ω pour tout k et que la suite $\{\|x^k-y^k\|\}$ ne possède pas de sous-suite convergente vers 0. Dans cette situation, le résultat de Canon et Cullum (1968) est encore vrai:

pour tout $\alpha,\varepsilon > 0$, $F(x^k) - F(x^*) \geq 1/k^{1+\varepsilon}$

pour un nombre infini d'indices k. Malheureusement, la condition 2) précédente est très contraignante.

Comme nous pouvions nous en douter, la suite $\{F(x^K)\}$ semble décroître très lentement près de l'optimum.

Nous donnerons maintenant une borne sur la convergence la plus lente de la méthode d'accumulation des contraintes. Le développement qui suit est basé essentiellement sur un résultat de Wolfe (1970).

Nous supposons que F est continue, continuement différentiable 2 fois, convexe, qu'elle possède une constante positive "a/2" constituant une borne inférieure sur toutes les valeurs propres du Hessien $\nabla^2 F(x)$, et une constante positive "A/2" constituant une borne supérieure sur toutes les valeurs propres du Hessien.

Lorsque nous ne sommes pas à l'équilibre, nous pouvons démontrer par induction que

$$(C(x^j),\ x^j-x^k) > 0 \quad \text{pour tout } j < k.$$

Cela nous amène à imposer l'hypothèse supplémentaire suivante:

"$\exists M > 0$ finie tel que

$$\max_{u\in\Omega} (C(x^j),\ x^j-u) \leq M(C(x^j),\ x^j-x^k) \text{ pour tout } j < k" \qquad (2.2.3.2)$$

Celle-ci est vérifiée en pratique.

Dès lors, nous pouvons écrire à partir des 2 relations précédentes:

$$\|C(x^j)\| > 1/2\ a\|x^j-x^*\| \text{ pour tout } j \qquad (2.2.3.3)$$

De plus, on a grâce à la convexité de F:

$$F(x^j) - F(x^*) < (C(x^j), x^j - u_{x^j}) \qquad (2.2.3.4)$$

où u_{x^j} est un plus court chemin associé à $C(x^j)$.

On obtient d'après (2.2.3.2) et (2.2.3.4) que

$$F(x^j)-F(x^*) \le M(C(x^j),\ x^j-x^k) \text{ pour tout } j < k$$
$$\le M\|C(x^j)\| \cdot \|x^j-x^k\| \text{ pour tout } j < k.$$

Or par hypothèse, nous obtenons la relation

$$\|C(x^j)\|^2 \le A(F(x^j)-F(x^*))$$

de sorte que

$$\|C(x^j)\| \le M\cdot A\cdot |x^j-x^k| \text{ pour tout } j < k. \qquad (2.2.3.5)$$

En combinant (2.2.3.3) et (2.2.3.5), on obtient:

$$\|x^j-x^k\| > d\|x^j-x^*\| \text{ pour tout } j < k,$$

où $d = a/2\cdot M\cdot A$. En utilisant ce résultat, et en réordonnant la suite $\{x^k\}$ de façon à obtenir la suite $\{\bar{x}^k\}$ telle que les valeurs $\|\bar{x}^k-x^*\|$ sont non-croissantes, P.Wolfe (1970) obtient:

$$\|\bar{x}^k-x^*\| < 2\|\bar{x}^1-x^*\| \cdot 2^{-k/N}$$

où

$N = (1+4/d)^n$, n = nombre de variables.

Ainsi, ces réarrangement assure la convergence de la suite $\{\bar{x}^k\}$ vers une série géométrique au taux de $2^{-1/N}$. Nous ne pouvons pas affirmer toutefois que les distances $\|x^k-x^*\|$ sont dominées par une série géométrique. Nous pouvons indiquer que la convergence de la suite $\{x^k\}$ ressemble à celle d'une série

géométrique à un taux de

$$-1/(1+8\cdot M\cdot A/a)^{N}$$

2

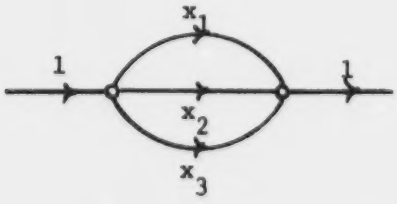
Dans le prochain chapitre, nous examinerons expérimentalement le comportement des suites $\{F(x^k)\}_{k=1}^{\infty}$, $\{\|x^k-x^*\|\}_{k=1}^{\infty}$. De plus, sachant que les méthodes de coupe convergent lentement près de l'optimum [Dinkel et al., 1977; Hartley et Hocking, 1963; etc.] et que le nombre de coupes générées peut devenir impraticable, nous verrons différentes techniques pour pallier à ces inconvénients.

Mais auparavant, concentrons-nous sur les liens qui existent entre notre approche et celle de Frank-Wolfe.

2.2.4 Parallèle avec l'algorithme de Frank et Wolfe.

La méthode d'accumulation des contraintes proposée dans ce texte pour résoudre le problème de la recherche d'un flot d'équilibre dans un réseau routier est très reliée à la méthode de Frank & Wolfe dans le cas "symétrique". L'approche de Frank & Wolfe correspond à éliminer toutes les coupes sauf la dernière à chaque itération.

Examinons d'abord si la méthode de Frank & Wolfe converge encore dans le cas asymétrique. Un contre-exemple a été donné par Magnanti (1982) dans le cas de fonctions de coûts strictement monotones:



$$C(x) = \begin{bmatrix} 4 & 4 & 1 \\ 1 & 4 & 4 \\ 4 & 1 & 4 \end{bmatrix} x + \begin{pmatrix} -3 \\ -3 \\ -3 \end{pmatrix} = Mx + q.$$

Les valeurs propres de la matrice M sont $\lambda_1 = 9$, $\lambda_2 = 3/2[1 \pm \sqrt{3}I]$.

La matrice M est définie positive si et seulement si la matrice symétrique $M + M^t$ est définie positive, ce qui est vrai si le déterminant de chacun de ses mineurs principaux est positif. Puisque ces déterminants valent respectivement 8, 39 et 162, M est définie positive et, alors, C(x) est strictement monotone.

Le vecteur de flot $x^* = (1/3, 1/3, 1/3)$ est l'unique solution d'équilibre.

En désignant par x^k = le point courant, u^k = une solution extrémale de Min $\{(C(x^k),u) \mid u \in \Omega\}$ et θ^k = le pas de descente, à l'itération k, l'algorithme de Frank-Wolfe nous donne:

$$\begin{cases} x^1 = (1,0,0),\ C(x^1) = (1,-2,1),\ u_{x^1} = (0,1,0),\ \theta^1 = 1\ . \\ x^2 = (0,1,0),\ C(x^2) = (1,1,-2),\ u_{x^2} = (0,0,1),\ \theta^2 = 1\ . \\ x^3 = (0,0,1),\ C(x^3) = (-2,1,1),\ u_{x^3} = (1,0,0),\ \theta^3 = 1\ . \\ x^4 = (1,0,0). \end{cases}$$

Nous obtenons un cycle.

Remarquons que $|\lambda_2| = 3 > 1$, $|\lambda_1| = 9 > 1$ où λ_1 et λ_2 sont les valeurs propres de M. De plus, Magnanti affirme qu'il serait intéressant de construire un exemple comme celui qui précède où la fonction de coût est non-négative sur Ω.

En voici un! Soit

$$M = \begin{bmatrix} \varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \end{bmatrix}, \quad q = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{ où } \epsilon > 0 \text{ est aussi petit que l'on veut.}$$

Les mineurs principaux de $(M + M^t)$ sont 2ε, $3\varepsilon^2$ et $4\varepsilon^3$. Les valeurs propres de $M + M^t$ sont ε, ε et 4ε.

On obtient le cycle suivant:

$$\begin{cases} x^0 = (1,0,0),\ C(x^0) = (\varepsilon,0,\varepsilon),\ u_{x^0} = (0,1,0),\ \theta^0 = 1 \\ x^1 = (0,1,0),\ C(x^1) = (\varepsilon,\varepsilon,0),\ u_{x^1} = (0,0,1),\ \theta^1 = 1 \\ x^2 = (0,0,1),\ C(x^2) = (0,\varepsilon,\varepsilon),\ u_{x^2} = (1,0,0),\ \theta^2 = 1 \\ x^3 = (1,0,0). \end{cases}$$

De plus, les valeurs propres de M sont toutes égales à 2ε ce qui est inférieur à 1. Enfin, $C(x) \geq 0$ pour tout $x \in \Omega$.

Nous regarderons maintenant l'algorithme de Frank & Wolfe comme une approximation linéaire d'un problème convexe différentiable.

En effet, réécrivons le problème Min-Max convexe:

$$\min_{x\in\Omega} g(x) \qquad (2.2.4.1)$$

où

$$g(x) = \max_{y\in\Omega} (C(y),x-y) \qquad (2.2.4.2)$$

Nous avons des raisons de croire que g est différentiable sur Ω presque partout. S'il en est ainsi, nous pouvons voir facilement que $(u_{y_x} - y_x)$ est une direction de descente de g au point x où

$$y_x = \arg g(x) \quad \text{et} \quad u_{y_x} = \arg \min_{z\in\Omega} (C(y_x), z).$$

Si nous voulons appliquer l'algorithme de Frank & Wolfe au problème (2.2.4.1 - 2.2.4.2), nous devons calculer $\nabla g(x) = C(y_x)$ ce qui implique bien entendu la détermination de y_x. En utilisant la version de l'algorithme de Frank & Wolfe correspondant au coût C(x) standard, on substitue le gradient $\nabla g(x)$ au point x courant, par le vecteur C(x), autrement dit nous approximons $C(y_x)$ par C(x). Or, cette approximation n'est pas toujours valable. Les exemples précédents sont là pour en témoigner.

Une "meilleure" approximation de y_x au point x s'obtient en calculant

$$z = x + \bar{\theta}(u_x - x)$$

où

u_x est une solution extrémale de Min $\{(C(x), u) \mid u \in \Omega\}$

$$\bar{\theta} = \arg \min_{\theta \in [0,1]} (C(x+\theta(u_x - x)), x+\theta(u_x - x) - x).$$

Mais, le choix de $d = (u_z - x)$ (où u_z est une solution extrémale de Min $\{(C(z), u) \mid u \in \Omega\}$) comme direction au point x dans l'algorithme de Frank-Wolfe n'est pas judicieux.

En effet, il se peut que le pas de Frank & Wolfe soit nul sans que l'optimum soit atteint. Des essais sur le réseau de Hull nous ont convaincus (voir chapitre 3).

De même, la suite $\{x^k + 1/k\ (u_{z^k} - x^k)\}_{k=1}^{\infty}$ ne converge pas.

Dans le chapitre suivant, nous décrirons l'implantation de la méthode d'accumulation des contraintes, les techniques utilisées pour économiser de l'espace-mémoire et pour accélérer la convergence de l'algorithme. Plusieurs essais sont effectués sur des réseaux de différentes tailles.

CHAPITRE 3

METHODE D'ACCUMULATION DES CONTRAINTES: RESULTATS DES CALCULS.

3.1 INTRODUCTION

A cette étape, nous voulons surtout mettre en évidence la convergence expérimentale de la méthode d'accumulation des contraintes en termes du nombre d'itérations requises pour atteindre la solution optimale. A la fin de ce chapitre, nous comparerons nos résultats avec ceux provenant d'algorithmes déjà connus.

Nous allons examiner en détail le design et les essais de procédures particulières pour des réseaux de moyennes et grandes tailles. Les réseaux de l'île de Hull (comportant 501 sommets, 798 arcs et 142 paires O/D) et de Winnipeg (comportant 1052 sommets, 2836 arcs et 147 centroïdes) seront utilisés à travers ce chapitre comme problèmes types. Nous fabriquerons aussi un réseau ayant des coûts asymétriques. Plusieurs petits problèmes ont aussi été résolus en appendice afin de ne pas surcharger le texte; ceux-ci ont souvent été résolus dans des travaux antérieurs par différents auteurs.

Il s'agit en premier lieu de regarder, en termes généraux, diverses approches de résolution des sous-problèmes linéaires

$$\text{Max } x_o$$
$$x_o + (a^i, y) \underset{y \in \Omega}{\leq} b \quad , \quad i = 1,2,\ldots,k \qquad (3.1.1)$$

où $a^i = C(x^i)$, $b = (C(x^i), x^i)$ et Ω est l'ensemble des flots réalisables, supposé borné, décrit par les équations de conservation des multi-flots. On peut supposer que x_o est bornée sans changer la solution du problème linéaire.

Les calculs les plus importants dans l'application de la méthode d'accumulation des contraintes proviennent de la résolution des sous-problèmes de la forme (3.1.1).

Il est clair que l'on doit traiter implicitement les équations de conservation des multi-flots, en nombre généralement très grand pour un problème de taille raisonnable, si l'on veut résoudre ces programmes linéaires de façon économique, aussi bien en quantité de mémoire d'ordinateur requise qu'en temps d'exécution. Dans cette perspective, deux approches mutuellement duales l'une de l'autre peuvent être considérées:

1) la dualisation des contraintes (3.1.1),
2) la décomposition primale de Dantzig-Wolfe (1963).

Si l'approche duale offre une plus grande flexibilité quant au choix des méthodes de révision des variables duales (méthodes des sous-gradients, méthode d'approximation tangentielle), elle a le désavantage de ne pas donner directement une solution primale réalisable. Pour des problèmes de grandes tailles, nous pensons que la méthode primale de Dantzig-Wolfe soit plus appropriée. La génération des colonnes se ramène ici essentiellement à la recherche de plus courts chemins dans un réseau, pour laquelle de nombreuses techniques très efficaces existent (voir par exemple Dial et al., 1979; Florian et al., 1979).

Lorsque la k+1 ième contrainte (3.1.1) est ajoutée pour former le nouveau sous-problème, possédant déjà une solution duale réalisable on pense immédiatement à utiliser l'algorithme dual du simplexe pour réoptimiser le programme maître courant de la décomposition de Dantzig-Wolfe. Cette combinaison non-triviale demande une étude plus poussée des structures particulières du problème. Nous adoptons plutôt ici une approche composite, parente à la méthode de pénalité qui consiste essentiellement à maximiser une fonction objectif composite:

$$\zeta = wx_o - z, \qquad 0 < w < 1$$

où -z est une variable artificielle ajoutée à la nouvelle k+1 ième contrainte afin que la solution optimale de l'étape précédente puisse être utilisée comme solution initiale pour le nouveau sous-problème. L'exposé de l'application de la méthode de décomposition de Dantzig-Wolfe couplée avec une méthode composite de réoptimisation est donné à la prochaine section.

La détermination de la nouvelle k+1 ième contrainte

$$H^{k+1}: (C(x^{k+1}), x^{k+1}-y) \geq x_o \qquad (3.1.2)$$

ne présente aucune difficulté particulière. En effet, étant donné x^k: le point courant, y^k: la solution optimale de (3.1.1), le calcul du nouveau point x^{k+1}:

$$x^{k+1} = x(\theta^{k+1}) = x^k + \theta^{k+1}\,(y^k - x^k),$$

donc de θ^{k+1} défini par

$$\theta^{k+1} = \arg\min_{0\le\theta\le 1} [(C(x(\theta)),\, x^k - y^k)]^2$$

se fait très simplement comme suit:

$$\theta^{k+1} = \begin{cases} 1 \text{ si } (C(y^k),\, x^k - y^k) > 0 \\ \bar{\theta} \text{ autrement,} \end{cases}$$

où

$$(C(x(\bar{\theta})), x^k - y^k) = 0. \qquad (3.1.3)$$

On peut ici appliquer la méthode de Newton pour calculer la racine $\bar{\theta}$ de (3.1.3). Noter qu'une valeur exacte de $\bar{\theta}$ n'est pas nécessaire, puisqu'il suffit d'une valeur positive de θ^k pour que la nouvelle contrainte H^{k+1} élimine la solution optimale courante.

Nous avons supposé jusqu'ici que toutes les contraintes H^i, $i = 1,2,\ldots,$ k sont conservées. En fait, on peut laisser tomber certaines contraintes non-actives des sous-problèmes subséquents sans altérer la convergence de la procédure globale. En adaptant un schéma général développé par Eaves et Zangwill (1971), on peut éliminer la contrainte H^i des sous-problèmes à partir de l'itération k+1 si:

a) H^i est non-active au point optimal du k ième sous-problème, et

b) $x_o^k < x_o^i - \varepsilon x_o^i$,

où ε est un scalaire positif arbitrairement choisi. Comme nous pouvons choisir ε aussi petit que nous voulons, en pratique on peut enlever les contraintes inactives après chaque itération lorsqu'il y a diminution stricte de x_o.

3.2 IMPLANTATION DE L'ALGORITHME: METHODE DE DANTZIG-WOLFE

Considérons le problème linéaire

$$\text{Max } x_o \tag{3.2.1}$$

sujet à:

$$(C(x^i),\ x^i - u) \geq x_o \qquad i = 1,2,\ldots,k$$
$$u \in \Omega$$

que l'on doit résoudre successivement pour $k = 1,2,\ldots$; Ω est l'ensemble des flots réalisables, décrit par les équations de conservation des multi-flots. On suppose Ω borné. Il est nécessaire de faire appel à la méthode de décomposition de Dantzig-Wolfe pour éviter le traitement explicite des équations de conservation des multi-flots, généralement en nombre très grand.

Soit y^j, $j \in J$ les points extrémaux de Ω en nombre fini, en exprimant u en fonction des y^j on obtient le problème équivalent:

$$\text{Max } x_o \tag{3.2.2}$$

sujet à $\quad \sum_j \lambda_j = 1$

$$x_o + \sum_j A_i^j \lambda_j + e_i = b_i,\ i = 1,2,\ldots,k$$
$$\lambda_j \geq 0$$

où $A_i^j = (C(x^i), y^j)$,
$b_i = (C(x^i),\ x^i)$,

et e_i est la variable d'écart de la coupe i.

A l'étape k, nous avons une solution optimale $(x_o^{k-1},\ \lambda^{k-1},\ e^{k-1})$ du problème (3.2.2) sans la k ième contrainte. Il serait maladroit de ne pas utiliser cette solution duale réalisable pour réoptimiser (3.2.2) à l'étape k, surtout qu'en s'approchant du flot d'équilibre, le point (x_o, λ) ne changent que légèrement d'une étape à l'autre. La réoptimisation à l'aide de la méthode duale du simplexe n'étant pas pratique, à cause de la génération des colonnes qui est implicite dans la décomposition de Dantzig-Wolfe, nous devons appliquer ici une méthode composite avec addition d'une variable artificielle à la k ième contrainte. On obtient alors le problème suivant:

$$\begin{aligned}
&\text{Max } \zeta = -z + wx_o && (3.2.3)\\
&\sum_j \lambda_j = 1\\
&x_o + \sum_j A_i^j \lambda_j + e_i = b_i \qquad i = 1,2,\ldots,k-1\\
&-z + x_o + \sum_j A_k^j \lambda_j = b_k\\
&z \geq 0, \quad \lambda_j \geq 0
\end{aligned}$$

où w est une pondération positive arbitrairement choisie; l'obtention d'une solution primale réalisable à partir de $(x_o^{k-1}, \lambda^{k-1})$ ne se fera donc pas de façon totalement indépendante de l'objectif initial (maximiser x_o). Noter que l'emploi de la variable d'écart e_k n'est pas nécessaire.

Nous allons maintenant examiner les diverses étapes de l'application de la forme révisée de la méthode du simplexe pour la réoptimisation de (3.2.2) à l'aide de la formulation (3.2.3). Considérons tout d'abord la suppression des contraintes non-actives à la fin de l'étape (k-1).

SUPPRESSION DES CONTRAINTES NON-ACTIVES A LA FIN DE L'ETAPE (k-1)

Soit B' la base optimale à la fin de l'étape (k-1), si la ième variable d'écart est strictement positive, la ième contrainte est non-active; en supprimant cette contrainte, on doit enlever la ième ligne et disons la jème colonne de la base B'. Puisque la jème colonne de B' a un seul élément égal à l'unité, on peut déduire immédiatement l'inverse de la nouvelle base réduite en supprimant simplement la jème ligne et la ième colonne de B'^{-1}.

En supprimant toutes les contraintes non-actives, en réorganisant les lignes et les colonnes si nécessaire, on obtient une base réduite B et son inverse B^{-1} d'ordre $M \leq k$. Cette réduction est nécessaire afin de conserver des bases d'ordre raisonnable.

INITIALISATION DE LA METHODE DU SIMPLEXE

Une solution initiale réalisable pour (3.2.3) est

$$z = x_o^{k-1}, \qquad x_o = x_o^{k-1},$$

$$\lambda_j = \lambda_j^{k-1}, \; j \in J,$$

$$e_i = 0, \; i \in I,$$

où I est l'ensemble des indices des contraintes qui demeurent.

La base $\tilde{B}$ est:

$$\tilde{B} = \begin{array}{|c|c|} \multicolumn{1}{c}{(x_o,\lambda)} & \multicolumn{1}{c}{z} \\ \hline B & 0 \\ \hline \beta & -1 \\ \hline \end{array}$$

où $\beta = (1, A_k^{j_1}, A_k^{j_2}, A_k^{j_3}, \ldots, A_k^{j_m})$

et rappelons que

$$A_k^j = (C(v^k), y^j).$$

L'inverse $\tilde{B}^{-1}$ s'obtient directement de B^{-1} comme suit:

$$\tilde{B}^{-1} = \begin{array}{|c|c|} \hline B^{-1} & 0 \\ \hline \beta B^{-1} & -1 \\ \hline \end{array}$$

Pour alléger la notation posons

$$v = (x_o, \lambda_{j1}, \lambda_{j2}, \ldots, z, e_1, e_2, \ldots)^t$$

et désignons par $J^{\tilde{B}}$ l'ensemble des indices des variables de base et $v^{\tilde{B}}$ les variables de base.

CALCUL DES MULTIPLICATEURS DE LAGRANGE

Les variables duales $(\mu:\Pi)$ vérifient:

$$(\mu:\Pi)\ \tilde{B} = [w,0,0,\ldots,0,-1],$$

w et -1 étant respectivement les coefficients sur la fonction objectif de x_o et z.

$$(\mu:\Pi) = (wB_1^{-1} - \beta B^{-1} : 1)$$

où B_1^{-1} est la première ligne de B^{-1} tandis que βB^{-1} a été calculé à l'étape précédente.

RECHERCHE DE LA COLONNE A FAIRE ENTRER DANS LA BASE

La variable x_o n'est pas astreinte à être non-négative et peut toujours rester dans la base. Dès que la variable z sort de la base on peut l'ignorer pour la suite des calculs. Il faut seulement considérer les variables λ_j et les variables d'écart.

Le coût réduit d'une variable d'écart e_i est $-\Pi_i$;

posons donc:

$$\Pi_s = \min_i \Pi_i .$$

Si $\Pi_s < 0$, alors e_s est candidat à l'entrée de la base et nous passons à la phase de calcul suivante.

Le coût réduit d'une variable λ_j est

$$\bar{c}_j = -\mu - \Pi A^j .$$

La plus grande valeur de $\bar{c}_j$ est déterminée par la résolution du sous-problème:

$$F(y^s) = \min_{y \in \Omega} (\hat{c}, y) + \mu$$

où

$$\hat{c}_a = \sum_i \pi_i c_a(x^i)$$

qui se ramène, on le sait, à un problème de plus courts chemins dans un réseau. Notons que tous les π_i sont non-négatifs; par conséquent, si les coûts de transport sont non-négatifs, il n'existe aucun circuit négatif sur le réseau.

Si $F(y^s) < 0$, A^s est la colonne à faire entrer dans la base et la procédure continue. Si z n'est pas nul à l'optimum, on doit diminuer la pondération de w et continuer le processus tandis que si z est égale à 0, le problème est résolu.

RECHERCHE DE LA COLONNE A FAIRE SORTIR DE LA BASE

Afin de déterminer la variable qui devra sortir de la base, il faudra calculer la représentation de la colonne entrante en termes des colonnes de la base, soit:

$$\bar{D}^s = \tilde{B}^{-1} \left(\frac{1}{A^s} \right)$$

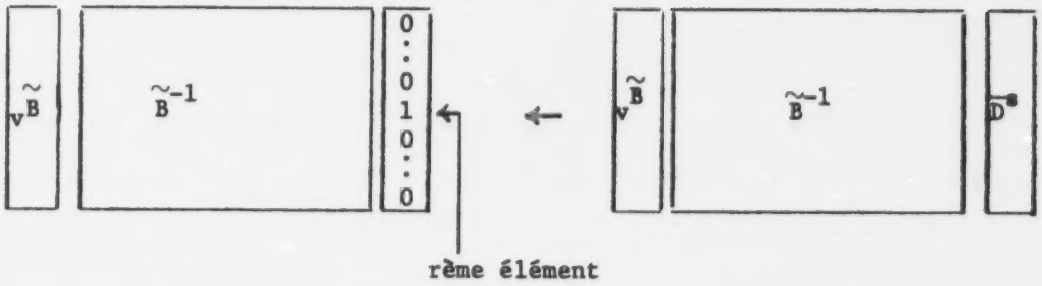
où $A^s_i = (C(x^i), y^s)$.

Le choix de la variable $v^{\tilde{B}}_r$ qui va sorrir de la base se fait de façon usuelle, c'est-à-dire,

$$\frac{v^{\tilde{B}}_r}{\bar{D}^s_r} = \min_{\bar{D}^s_i > 0} v^{\tilde{B}}_i / \bar{D}^s_i$$

PIVOTAGE

On met à jour l'inverse de la base et la valeur des variables de base à l'aide d'une procédure de pivotage illustrée par le schéma suivant:



Par la suite, on retourne à la phase de calcul des multiplicateurs.

Dans la prochaine section, nous examinerons les structures de donnée, l'espace-mémoire exigé, ainsi que les modules de programmation qui entrent en jeu dans l'implantation de l'algorithme d'accumulation des contraintes.

3.3 STRUCTURES DE DONNEE; ESPACE-MEMOIRE ET MODULES DE PROGRAMMATION ENTRANT EN JEU

Posons

M=# d'arcs,

N=# de sommets,

P=# de paires O/D,

R=# maximum de coupes.

Nous énumérons maintenant les entrées-sorties de notre algorithme de coupe ainsi que l'espace-mémoire exigé.

NCENT(INPUT):	nombre de centroïdes	1
NNODE(INPUT):	nombre de sommets	1
NLINK(INPUT):	nombre de liens	1
NPOD(INPUT) :	nombre de paires O/D	1
NOLAB(INPUT):	vecteur de pointeurs sur les liens. NOLAB(I) pointe au premier lien ayant comme noeud origine I	N+1
NO(INPUT) :	vecteur contenant les noeuds origine des liens. NO(L) donne le noeud origine du lien L.	M
ND(INPUT) :	vecteur contenant les noeuds destination des liens. ND(L) donne le noeud destination du lien L	M

COST(INPUT):	matrice contenant les coefficients pour les fonctions volume-délai sur les arcs	?
LINIT(INPUT):	vecteur contenant les noeuds origine des paires O/D	p
LTERM(INPUT):	vecteur contenant les noeuds destination des paires O/D	p
DEMANDE(INPUT):	vecteur de demande associé aux paires O/D. DEMANDE (I) donne la demande de la ième paire O/D	p
MAXCOUP(INPUT):	le nombre maximum de coupes qui peuvent être retenues à une itération donnée	1
EPSILON(INPUT):	paramètre prenant en charge les erreurs d'arrondi et/ou de troncation qui surviennent au cours du processus de calcul	1
MAXITER(INPUT):	nombre maximum d'itérations de l'algorithme	1
GAPMIN(INPUT) :	valeur qui donne le critère d'arrêt, correspond au degré de précision de l'objectif du problème linéaire ayant un nombre infini de contraintes	1
OMEGA(INPUT) :	paramètre de la fonction objectif composite du problème linéaire	1
PARIMP(INPUT) :	paramètre d'impression	1
ISER(INPUT) :	rejet ou non des contraintes actives	1
LPAS(INPUT) :	type de pas utilisé	1
NOITE(OUTPUT) :	numéro de l'itération	1
TIME(OUTPUT) :	temps(cumulé) d'exécution de l'algorithme (les opérations initiales d'entrée-sortie sont exclues)	1
XOK(OUTPUT) :	valeur de l'objectif du problème linéaire à l'optimum à l'itération courante	1
FY(OUTPUT) :	valeur optimale du problème $\max_{y\in\Omega} (C(x^k), x^k - y)$ où x^k est la solution courante	1
THETA(OUTPUT) :	valeur du pas à l'itération courante	1

IR(OUTPUT)	:	nombre de coupes en présence à l'itération courante	1
VN(OUTPUT)	:	valeur de la norme $\|x^k-x^{k-1}\|$	1
LP(OUTPUT)	:	nombre d'itérations de l'algorithme connu sous le nom de G.U.B. (Generalized Upper Bounding) dû à Dantzig et Van Slyke (1967)	1
LPEC(OUTPUT)	:	nombre de changements de variables d'écarts	1
VEQ(OUTPUT)	:	valeur approximative de $\|x^k-x^*\|$	1

L'espace de travail exige aussi $5m+3n+r^2+7r$. Enfin, 2 fichiers sur mémoire à accès sélectif ("random file") contiennent respectivement les points extrémaux "actifs" de Ω et les vecteurs $C(x^i)$ associés aux coupes actives.

Nous donnons maintenant en détail les principaux modules qui entrent en jeu dans l'implantation de l'algorithme.

Bloc des routines-fonctions			Espace-mémoire (en mot de 64 bits)
AJUSTX,AJUSTY	:	permet la gestion des fichiers à accès sélectif	2x29
LOAD	:	résolution d'un problème de plus court chemin et stockage du flot tout-ou-rien en mémoire	75
INIT	:	initialisation de l'inverse de la base, des indices des colonnes de base et des indices des lignes de base, du rang de la base, de la fonction objectif et de la valeur des variables de base	46
PURGE	:	purge les contraintes non-actives	168
NEWINV	:	calcul du nouvel inverse au début de chaque itération	76
OPT	:	évaluation de la solution optimale du problème linéaire et de la valeur de la fonction objectif x_o^k	89

GENERK	: permet de générer une nouvelle ligne dans la base	91
IMPRESU	: impression des résultats à chaque itération	
NEWTON	: calcul du pas à chaque itération, peut changer selon les problèmes à résoudre	?
FOBJ	: dans le cas de fonctions de coûts "symétriques", elle permet d'évaluer la fonction objectif de Beckmann au point courant	109
UCOST	: Sous-routine calculant pour un lien la fonction volume-délai	96
PARGUB, RDNET, RDOD	: permet de lire les paramètres d'entrée de l'algorithme, les informations propres au réseau et les données pour chaque paire O/D respectivement.	133 187 48*
IMPNET, IMPOD	: impression des données du réseau et de la matrice O/D	126,69*
MINTREE	: sous-routine de plus court chemin	126

* Ces valeurs sont celles trouvées lors de la résolution du problème du réseau de Hull.

Le programme principal qui contient en plus du processus de calcul proprement dit, les appels d'entrée-sortie et la préparation des variables d'impression exige 987 mots mémoires.

Considérons maintenant les résultats obtenus à partir d'essais sur les réseaux de Hull et Winnipeg.

3.4 RESULTATS DES CALCULS SUR LES RESEAUX DE HULL ET DE WINNIPEG

Nous avons résolu le problème de la recherche d'un flot d'équilibre sur les réseaux de Hull et de Winnipeg. Ces réseaux de grande taille comprennent respectivement 501 sommets, 798 arcs, 23 centroïdes et 1052 sommets, 2836 arcs et 147 centroïdes.

Ces deux problèmes possèdent des fonctions de coût "symétriques". Une description complète de chacun de ces réseaux, de leur matrice O/D et des paramètres de leur fonction de coût est contenue dans les rapports techniques de D. Branston et al. (1975) et Project EMME (1978).

Dans la méthode d'accumulation de contraintes développée antérieurement, nous nous sommes aperçus que lorsque le nombre d'itérations devient grand, le calcul exact de la solution du sous-problème de programmation linéaire devient coûteux. Cela nous amène à développer un nouveau schéma de calcul où une solution approchée du problème linéaire est obtenue. Puisque le processus d'élimination des contraintes non-actives est basé sur une diminution stricte suffisante de l'objectif du problème linéaire à l'optimum, nous devons penser à une façon différente d'éliminer les contraintes indésirables.

Dans le cas symétrique, nous pouvons nous baser sur la diminution de l'objectif du problème de Beckmann (1956). En effet, pour déterminer une direction de descente de l'objectif de Beckmann au point courant x^k, il suffit de déterminer une solution réalisable (non-nécessairement optimale) du sous-problème linéaire à l'itération k telle que l'objectif x_o soit strictement positif.

Examinons maintenant les résultats obtenus sur les réseaux de Hull et de Winnipeg. Nous reviendrons plus tard au cas asymétrique.

Considérons d'abord le réseau de Hull. Le tableau (3.4.1) contient les résultats qui s'y rattachent.

Puisque les fonctions de coût sont "symétriques", nous nous sommes servis du critère d'approximation suivant: en cherchant à résoudre le sous-problème de programmation linéaire à partir d'une méthode composite avec addition d'une variable artificielle z à la dernière contrainte (section 3.2), après la sortie de la variable z de la base, nous arrêtons les calculs dès que l'objectif du problème linéaire devient strictement positif. La génération de la coupe suivante selon la formulation de la section 3.2 permet alors d'éliminer la solution courante du sous-problème linéaire. En procédant ainsi, nous sommes assurés d'une diminution stricte de l'objectif de Beckmann. Le calcul du pas permet la meilleure amélioration possible dans la direction obtenue.

En examinant le tableau (3.4.1), nous voyons que le nombre de coupes conservées n'a jamais dépassé 6; en moyenne, nous avons conservé 3.81 coupes sur les 200 itérations effectuées. Aussi, à chaque itération, nous avons résolu au plus un problème de plus court chemin.

Comme prévu, la convergence de l'algorithme s'avère très lente près de l'optimum. Après la 50 ième itération, la diminution de la fonction objectif n'est plus sensible. Dans la prochaine section, nous analyserons différentes techniques d'accélération de la convergence.

Nous avons considéré aussi l'alternative qui consiste à effectuer plus de pivots après la sortie de la variable z mais le coût (temps d'exécution) était beaucoup plus élevé.

Enfin, dans le but de mesurer la profondeur des coupes générées et de pouvoir comparer possiblement celles-ci avec d'autres types de coupes, nous avons calculé une solution approximative de x^* (soit $\bar{x}$ la solution obtenue par notre algorithme à la 300 ième itération où la valeur du $GAP_{300} = 12.49$ et celle de la fonction objectif de Beckmann est égale à 34808.80). Dans le tableau 3.4.1, nous avons d'abord examiné la norme euclidienne $\|x^k-\bar{x}\|$. Puis, nous avons calculé le point le plus proche de $\bar{x}$ sur le dernier hyperplan de coupe ajouté à cette itération, c'est-à-dire

$$\begin{aligned} &\min\ 1/2\ \|x-\bar{x}\|^2 \\ \text{sujet à}\quad &(C(x^k),x^k-u) = (C(x^k),x^k-u^k) \\ &\qquad u\in R^M. \end{aligned}$$

En représentant la norme $\|x^k-\bar{x}\|$ vs le nombre d'itérations k (graphique 3.4.1), nous avons remarqué que la courbe avait la forme d'une exponentielle. En se servant de la théorie des moindres carrés, nous avons voulu ajuster cette courbe avec une forme connue. Malheureusement, cela s'avère difficile; il faudrait sectionner la courbe (voir graphique 3.4.1). Nous pouvons dire toutefois que cela ressemble à un taux de convergence linéaire ou d'ordre 1 d'après les essais effectués.

Itération k	Temps cumulé (sec.)	Gap courant (GAP_{k-1})	Nombre de coupes conservées	Norme $\|x^k - x^{k-1}\|$	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins résolus	Valeur de la fonction objectif du problème de Beckmann à x^k.
0	0,796	-	-	-	-	-	37 590,31
1	2,235	18 011,39	1	1 041,21	1	1	36 287,98
2	3,537	7 537,19	2	964,84	2	1	35 545,07
3	4,858	1 823,36	2	465,37	2	1	35 374,53
4	6,051	1 538,92	2	366,15	2	1	35 277,72
5	7,416	1 078,68	3	226,64	2	1	35 228,59
6	8,939	871,69	4	755,73	2	1	35 058,27
7	9,455	1 411,18	4	239,17	2	0	35 008,01
8	10,839	588,77	3	205,31	3	1	34 978,25
9	12,211	486,38	3	201,00	2	1	34 953,28
10	13,435	406,23	3	390,21	3	1	34 902,34
15	19,768	204,02	4	81,37	4	1	34 854,70
20	24,588	96,69	4	2,58	2	0	34 840,72
30	34,556	65,74	3	25,32	2	1	34 823,50
40	46,354	73,06	4	34,17	2	1	34 816,04
100	117,014	19,01	3	3,21	2	1	34 810,23
200	222,222	9,01	3	2,00	2	1	34 809,19

Tableau 3.4.1: Résultats obtenus avec le réseau de Hull à partir de la méthode d'accumulation des contraintes (résolution approximative de chaque sous-problème linéaire): cas symétrique

Légende: x^k = solution courante de la méthode d'accumulation des contraintes,

GUB signifie "Generalized Upper Bounding" (voir section 3.2)

$GAP_k = \underset{y \in \Omega}{\text{Max}} \; (C(x^k), x^k - y)$.

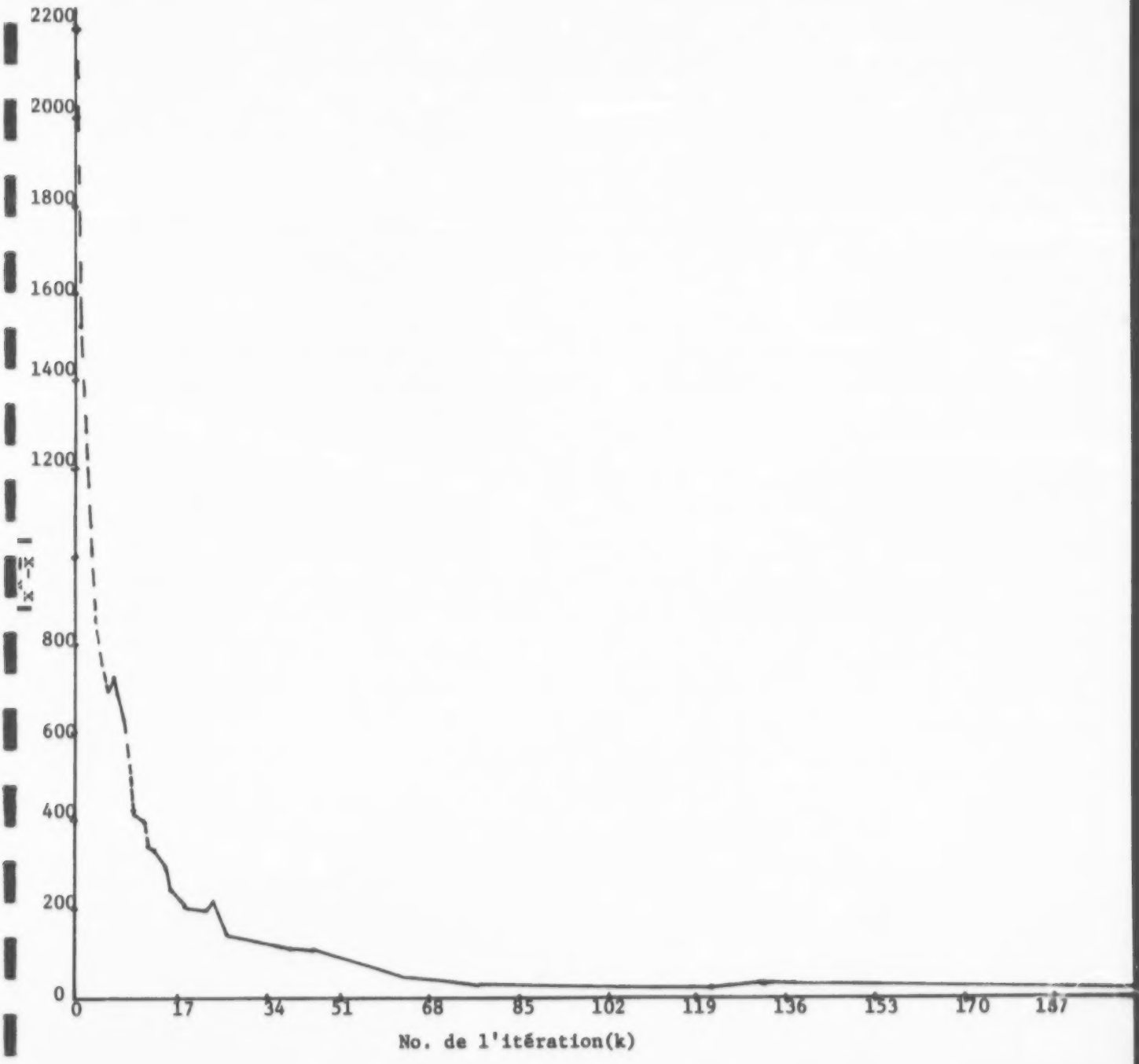
Itération k	Valeur de la norme $\|x^k-\bar{x}\|$	Valeur de $\|\tilde{x}_k-\bar{x}\|$ où $\tilde{x}_k$ représente l'optimum du problème quadratique $\text{Min } \frac{1}{2}\|x-\bar{x}\|^2$ sujet à $(C(x^k),u) = \min_{u\in R^M}(C(x^k),u^k)$
0	2 217,077	1,86465
1	1 532,472	0,83283
2	1 230,291	0,27122
3	1 014,561	0,11869
4	859,083	0,09812
5	758,706	0,01609
6	698,868	0,01629
7	734,804	0,00690
8	692,226	0,01045
9	638,819	0,01740
10	488,852	0,03264
15	338,651	0,00375
20	224,765	0,00164
30	148,183	0,00176
40	126,209	0,00165
100	31,881	0,00024
200	16,915	0,00009

Tableau 3.4.1 (suite)

Légende: $\bar{x}$ = solution approximative de x^*,

u^k = solution approximation du sous-problème linéaire à l'itération k.

Graphique 3.4.1 - La norme $\|x^k - \bar{x}\|$ vs le nombre d'itérations k.



Nous remarquons aussi d'après la dernière colonne du tableau (3.4.1) que la suite $\{\|\tilde{x}_k - \bar{x}\|\}_0^{\infty}$ se rapproche de zéro très rapidement dans les premières itérations pour ensuite progresser plus lentement dans les itérations subséquentes. Ceci représente bien le comportement de cet algorithme.

Nous proposons maintenant d'examiner le comportement de notre méthode d'accumulation des contraintes lors de la résolution du problème d'équilibre sur le réseau de Winnipeg. Cette fois aussi, nous avons en main la formulation équivalente de Beckmann puisque les fonctions de coût sont "symétriques". Nous procédons comme précédemment pour déterminer une solution approximative du problème linéaire à chaque itération.

Les remarques précédentes concernant le nombre de coupes emmagasinées, le nombre de problèmes de plus courts chemins, etc., demeurent vraies même pour ce réseau.

Le comportement de cet algorithme dans le cas de problèmes "symétriques" est excellent tant du point de vue du temps de calcul que de l'espace mémoire exigé.

Nous considérons maintenant le cas général où les fonctions de coûts sont "asymétriques". Nous construisons artificiellement un tel problème en se basant sur le réseau de Hull. Sachant que le Jacobien de la fonction de coût ("symétrique") pour le problème de Hull est diagonale et que la somme de deux opérateurs monotones est aussi monotone, nous posons

$$\tilde{C}(x) = C(x) + 10^{-6}A$$

où

$C(x)$ est la fonction de coût "symétrique",

$$A = (a_{ij})_{i,j=1,2,\ldots,M}$$

$$a_{ij} = \begin{cases} 1 & \text{si } i=j \\ 1/10 & \text{si } j = i+1 \\ 0 & \text{sinon} \end{cases}$$

Itération k	Temps cumulé (sec.)	Gap courant (GAP_k)	Nombre de coupes conservées	Norme $\|x^k - x^{k-1}\|$	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins résolus	Valeur de la fonction objectif du problème de Beckmann à x^k
0	24,226	3 038 533,49	-	-	-	-	1 391 948,84
1	51,482	957 525,76	1	14 409,79	1	1	1 071 078,62
10	267,811	59 742,31	4	1 636,92	3	1	911 548,20
20	487,048	19 320,60	3	1 104,69	2	1	896 575,83
30	641,622	15 123,60	3	1 007,61	3	1	893 008,93
40	833,779	7 749,08	4	53,01	2	0	891 532,76
50	1 021,580	7 697,11	4	265,87	2	0	890 657,97
60	1 252,221	3 606,83	3	250,15	2	1	890 125,44
70	1 480,009	4 358,94	5	269,04	3	1	889 718,26
80	1 690,685	4 051,87	4	97,01	3	1	889 556,32
90	1 918,378	2 377,17	4	110,63	3	1	889 353,54
100	2 146,450	1 924,95	3	120,94	2	1	889 257,97
120	2 542,591	1 592,25	4	120,65	2	1	889 166,40
140	2 996,977	2 221,01	3	115,77	2	1	889 121,73

Tableau 3.4.2: Résultats obtenus avec le réseau de Winnipeg à partir de la méthode d'accumulation des contraintes (résolution approximative de chaque sous-problème linéaire): cas symétrique

Légende: x^k = solution courante de la méthode d'accumulation des contraintes

GUB signifie "Generalized Upper Bounding" (voir section 3.2)

$GAP_k = \underset{y \in \Omega}{\text{Max}} \; (C(x^k), x^k - y)$.

M-# d'arcs, (certaines lignes de A sont nu' es pour certains arcs "dummy") de sorte que $\tilde{C}(x)$ est "asymétrique" et strictement monotone sur Ω.

Nous appliquons de manière intégrale notre méthode d'accumulation des contraintes à ce problème. Chaque sous-problème linéaire est résolu de façon exacte et les contraintes non-actives sont éliminées à chaque itération (Eaves et Zangwill, 1971).

Le tableau 3.4.3 contient les résultats obtenus en considérant le réseau de Hull avec les fonctions de coûts $\tilde{C}(x)$. On remarque immédiatement le coût élevé encouru dans la résolution de chaque sous-problème linéaire.

Par conséquent, nous devons examiner différentes stratégies pour pallier à cet inconvénient. La difficulté majeure provient du fait qu'il n'existe plus de formulation de Beckmann équivalente. En résolvant chaque sous-problème linéaire de façon approximative comme précédement, nous n'avons plus en main une suite monotone décroissante. D'où, à une étape donnée, nous pouvons difficilement caractériser la précision de la solution courante.

Nous avons reporté les résultats de cette heuristique dans le tableau 3.4.4. Nous nous sommes aperçus que cette approche peut du moins déterminer une "bonne" solution réalisable de départ. Nous avons aussi tracé la courbe de GAP_k (la valeur du "GAP standard" (Hearn,82) au point courant x^k) vs k (no. de l'itération) (voir graphique 3.4.2); celle-ci s'approche de zéro très rapidement dès les premières itérations.

Nous allons maintenant considérer deux nouvelles approches convergentes ayant comme objectif d'éviter le calcul exact de la solution de chaque sous-problème linéaire. La première approche consiste à résoudre exactement le sous-problème linéaire après r, 2r, 3r,... itérations (r est fixé à priori), et s'assurer d'une décroissance suffisante de l'objectif du problème linéaire aux itérations r, 2r,... pour permettre l'élimination des contraintes inactives. Par conséquent, cette technique de calcul est encore une méthode d'accumulation des contraintes où à chaque itération "extérieure", nous ajoutons r (>0) coupes au-lieu d'une seule.

Itération k	Temps cumulé (sec.)	Gap courant (GAP_{k-1})	Nombre de coupes conservées	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins résolus à l'étape k	Valeur exacte de l'objectif du sous-problème linéaire à l'optimum (x_o^k)
0	0,808	-	-	-	-	-
1	2,435	21 448,50	1	1	1	21 448,50
2	4,507	11 302,32	2	2	1	11 302,32
3	6,536	2 993,50	2	2	1	2 993,50
4	10,441	2 085,56	2	5	4	1 814,73
5	16,662	1 878,65	3	9	7	1 476,85
6	22,242	1 408,76	3	8	6	1 160,62
7	27,819	1 355,32	3	8	6	813,25
8	36,193	1 158,82	3	12	10	652,70
9	46,083	965,72	4	16	12	576,80
10	51,657	444,76	4	10	6	359,95
20	259,913	210,45	8	51	38	109,27
30	569,465	135,65	9	48	30	44,40

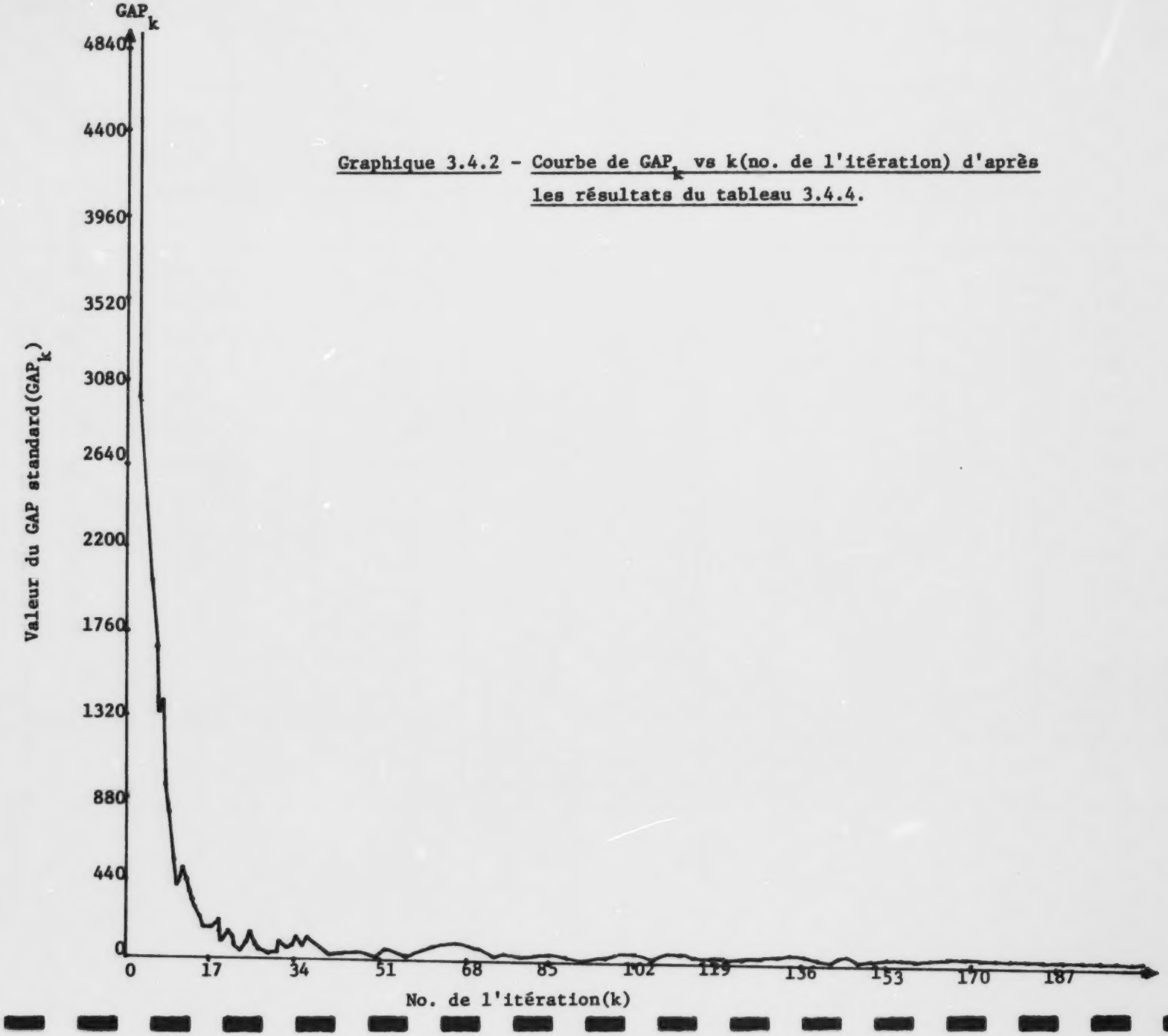
Tableau 3.4.3: Résultats obtenus avec le réseau de Hull à partir de la méthode d'accumulation des contraintes (résolution exacte de chaque sous-problème linéaire): cas asymétrique

Itération k	Temps cumulé (sec.)	Gap courant (GAP_{k-1})	Nombre de coupes conservées	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins résolus
0	0,811	-	-	-	-
1	2,436	21 448,50	1	1	1
2	3,816	11 302,32	2	2	1
3	5,184	2 993,50	2	2	1
4	6,402	2 085,56	2	2	1
5	7,813	1 637,06	3	2	1
6	9,364	1 314,48	3	2	1
7	10,781	1 327,79	3	2	1
8	12,193	970,14	3	2	1
9	13,452	784,44	3	2	1
10	14,870	547,57	3	2	1
15	20,287	328,54	3	2	1
20	27,158	245,44	3	3	1
30	39,568	109,91	3	3	1
40	52,257	57,90	4	4	1
100	119,340	36,51	3	2	1
200	226,296	28,97	3	2	1

Tableau 3.4.4: Résultats obtenus avec le réseau de Hull (cas asymétrique) à partir d'une heuristique qui consiste à résoudre approximativement chaque sous-problème linéaire.

Itération k	Temps cumulé (sec.)	Gap courant (GAP_k)	Nombre de coupes conservées à la fin de l'itération k après élimination des coupes non-actives	Valeur de l'objectif du sous-problème linéaire à l'étape k (x_o^k)	Nombre total d'itérations de GUB cumulé	Nombre de problèmes de plus courts chemins résolus cumulé
0	0,813	21 448,50	-	-	-	-
1	2,457	11 302,32	1	21 448,50	1	1
6	14,788	1 248,61	2	1 123,58	17	12
12	31,829	442,85	4	570,42	49	28
18	54,566	259,47	5	147,07	91	50
24	90,081	154,56	7	98,74	160	87
30	118,974	138,69	6	69,64	226	115
36	151,070	83,95	9	48,74	294	147
42	178,364	98,98	5	38,08	349	172
48	208,471	78,44	7	30,35	404	203
54	243,822	92,39	9	22,55	471	238
60	297,326	80,30	10	19,41	560	291

Tableau 3.4.5: Résolution exacte du sous-problème linéaire et rejet des contraintes non-actives aux itérations 1, 6, 12, 18,... (réseau de Hull: cas asymétrique)



Graphique 3.4.2 - Courbe de GAP_k vs k(no. de l'itération) d'après les résultats du tableau 3.4.4.

Aux itérations i·r+1, i·r+2, ..., i·r+r-1 pour tout i > 0, la solution approximative du sous-problème linéaire est telle que l'objectif x_o est strictement positif nous permettant ainsi d'éliminer la solution précédente.

Plusieurs essais sur ordinateur ont été effectués pour déterminer le meilleur choix de r. Lorsque r devient grand, la gestion des coupes devient plus coûteuse que l'économie obtenue en calculant peu de sous-problèmes linéaires de façon exacte.

Il semble que le choix de r(=6) soit un bon compromis entre ces deux pôles. Le tableau 3.4.5 représente les résultats obtenus.

On remarque immédiate la nette amélioration par rapport à l'algorithme initial (voir tableau 3. . Toutefois, le nombre de problèmes de plus courts chemins à résoudre est beaucoup trop élevé lorsque nous avons à résoudre exactement le sous-problème linéaire courant.

Nous suggérons maintenant une deuxième alternative de façon à améliorer l'algorithme de base (voir tableau 3.4.3). A cette fin, nous proposons la variante suivante de l'algorithme d'accumulation des contraintes:

ALGORITHME 3.4.1

Etape 0. Soit $x^o \in \Omega$ un point réalisable, $k \leftarrow 0$.

Etape 1. Trouver une solution approximative u^k du problème linéaire.

$$\text{Max } x_o$$

sujet à

$$(C(x^i), x^i - u) \geq x_o, \quad i = 0,\ldots,k \qquad (3.4.1)$$
$$u \in \Omega$$

telle que $x_o^k > 0$, à moins que la valeur de l'objectif de (3.4.1) à l'optimum soit nulle. Si tel est le cas, c'est terminé.

Etape 2. Soit

$$\theta^k = \arg\min_{\theta \in [0,1]} \{(C(x^k+\theta(u^k-x^k)), x^k-u^k)\}^2 ,$$

poser alors

$$x^{k+1} = x^k + \theta^k(u^k - x^k)$$

$x \leftarrow k+1$;

aller à l'étape 1.

Nous pouvons démontrer facilement que toutes les hypothèses du théorème de convergence de Zangwill (1969) sont encore satisfaites, de sorte que la limite de n'importe quelle sous-suite de $\{x^k, x_o^k\}_{k=0}^{\infty}$ vérifie le critère d'arrêt. Il nous reste maintenant à envisager la situation où le critère d'arrêt est satisfait.

PROPOSITION 3.4.1:

Si l'algorithme termine après k itérations, x^k est le flot d'équilibre, sinon il existe une sous-suite $\{x^k\}_{K_3}$ convergeant vers le flot d'équilibre.

Démonstration:

Soit x^* le flot d'équilibre. Si l'algorithme termine à l'étape k, alors (x_o^k, u^k) est une solution optimale de (3.4.1). Par conséquent, cela implique:

$$(C(x^i), x^i - x^*) = \min_{1 \le \ell \le k} (C(x^\ell), x^\ell - x^*) \le x_o^k.$$

D'autre part, la stricte monotonie de C permet d'écrire

$$(C(v), v - x^*) > 0 \text{ pour tout } v \neq x^* \qquad (3.4.2)$$

d'où,

$$x_o^k = 0 \rightarrow x^i = x^*.$$

Puisque $x_o^i \le (C(x^i), x^i - u^i) = (C(x^*), x^* - u^i) \le 0$ à l'itération i et que $x_o^\ell > 0$ pour tout $1 \le k-1$, on a:

$$k=i, \; x^k = x^i = x^*.$$

Considérons maintenant le cas où l'algorithme construit une suite infinie de points $\{x_o^k, x^k\}_o^{\infty}$, tous dans un compact, et $x_o^k > 0$. Puisque $x_o^k > 0$ alors $x^k \neq x^*$. Alors, pour n'importe quel sous-suite convergente $\{x_o^k\}$ vers 0, il existe une sous-suite K_1 convergente de

$$\{\min_{i=1,\ldots,k} (C(x^i), x^i - x^*)\}_{k \in K}$$

dont la limite est non-positive.

Construisons la sous-suite K_2 suivante:

$$K_2 = \{i_k \mid i_k = \arg \min_{i=1,\ldots,k} (C(x^i), x^i - x^*),\ k \in K_1\}.$$

D'après (3.4.2), il existe une sous-suite $K_3 \subset K_2$ telle que $\{x^k\}_{k \in K_3}$ converge vers x^*. □

Nous devons maintenant porter notre attention sur l'élimination des contraintes non-actives. Nous allons procéder comme suit: les contraintes sont conservées tant et aussi longtemps que le "gap standard" c'est-à-dire,

$$GAP(x) = \max_{y \in \Omega} (C(x), x-y)$$

n'a pas diminué strictement ($GAP(x^k) < B$ {le plus petit gap obtenu jusqu'à maintenant}, $B \in (0,1)$).

Les résultats sont reportés au tableau 3.4.6. Même si la suite $\{GAP(x^k)\}_{k=0}^{\infty}$ n'est pas monotone, nous avons quand même une bonne indication de la convergence de l'algorithme. De plus, on remarque que le nombre de coupes n'augmente pas de façon démesurée.

Nous aurions pû nous baser sur l'objectif x_o^k (non-nécessairement optimale) du problème de programmation linéaire à l'étape 1 de l'algorithme 3.4.1 au-lieu du "gap" $GAP(x^k)$, pour éliminer les contraintes non-actives. Mais, le nombre de coupes exigées est beaucoup trop élevé.

Plusieurs variantes possibles pourraient être examinées. Par exemple, une façon de procéder est de fixer une borne sur le nombre d'itérations nécessaires pour résoudre le sous-problème linéaire courant ou encore, de considérer une suite $\{\varepsilon_k\}_{k=0}^{\infty}$ monotone décroissante dont la limite est 0 de telle façon que la solution u^k obtenue à l'étape 1 de l'algorithme 3.4.1 soit ε_k-optimale. Ces dernières stratégies sont souvent utilisées dans les algorithmes de projection (Dafermos, 1980) ou les méthodes de diagonalisation (Florian, 1981).

Dans la prochaine section, nous avons examiné différentes stratégies pour améliorer la convergence de la méthode d'accumulation des contraintes. Nous avons considéré en particulier le problème de Beckmann (1956).

Itération k	Temps cumulé (sec.)	Gap courant (GAP(x^k))	Nombre de coupes conservées	Nombre total d'itérations de GUB cumulé	Nombre de problèmes de plus courts chemins résolus cumulé (1)
0	0,805	21 448,50	-	-	-
1	2,424	11 302,32	1	1	1
2	4,440	2 993,50	2	3	2
3	6,420	2 085,56	2	5	3
4	8,314	1 637,06	2	7	4
5	10,373	1 314,48	3	9	5
6	12,591	1 327,79	3	11	6
7	14,702	970,14	4	13	7
8	16,785	784,44	3	15	8
9	18,699	547,57	3	17	9
10	20,801	460,91	3	19	10
15	30,931	277,31	3	33	15
20	40,837	306,58	6	48	20
30	59,875	132,71	4	76	28
40	80,048	71,83	3	105	38
100	191,087	45,56	10	280	77

Tableau 3.4.6: Résultats obtenus en appliquant l'algorithme 3.4.1: le rejet des contraintes non-actives est relié à la décroissance du "GAP standard" (réseau de Hull, cas asymétrique).

(1) cela n'inclut pas la résolution d'un problème de plus court chemin à chaque itération afin de déterminer si les contraintes non-actives peuvent être éliminées.

3.5 TECHNIQUES D'ACCELERATION DE CONVERGENCE DE LA METHODE D'ACCUMULATION DES CONTRAINTES

L'objectif poursuivi dans cette section est de réduire le temps de calcul de la méthode d'accumulation des contraintes grâce à différentes stratégies développées ici.

Nous utiliserons une fois de plus le réseau de Hull (voir section 3.4) où les fonctions de coût sont "symétriques".

Une première stratégie pour améliorer la convergence de l'algorithme de base est de générer plus d'une coupe à chaque itération. En effet, nous pouvons soupçonner qu'en ajoutant plusieurs coupes d'intérêt à la fois au problème linéaire courant, le nombre de contraintes actives à chaque étape diminuerait et la convergence serait accélérée. Le schéma de calcul suivant est basé sur cette idée:

A l'itération k, soit (x_o^k, y^k) une solution optimale du problème

$$\max x_o$$

sujet à

$$(C(x^i), x^i - y) \geq x_o, \quad i \in I_k$$
$$y \in \Omega,$$

soit $\tilde{I}_k = \{i \in I_k \mid (C(x^i), x^i - y^k) = x_o^k\}$ l'ensemble des indices des contraintes actives,

soit $z^i = x^i + \theta^i (y^k - x^i)$, $i \in \tilde{I}_k$ où

$$\theta^i = \arg \min_{\theta \in [0,1]} ((C(x^i + \theta(y^k - x^i)), x^i - y^k))^2, \text{ pour tout } i \in \tilde{I}_k,$$

le nouveau problème à l'itération suivante s'écrit comme suit:

$$\max x_o$$
$$(C(x^i), x^i - y) \geq x_o, i \in \tilde{I}_k$$
$$(C(z^i), z^i - y) \geq x_o, i \in \tilde{I}_k$$
$$y \in \Omega.$$

Malheureusement, lorsque le nombre d'itérations augmente, le calcul des pas θ^i, $i \in \tilde{I}_k$ devient coûteux. De plus, la génération de plusieurs coupes à chaque itération selon le schéma de calcul précédent n'augmente pas la vitesse de convergence.

Il semble préférable d'utiliser une seule fois la solution optimale du sous-problème linéaire courant pour générer des coupes. Noter que cette conclusion n'est pas extraordinaire puisque d'autres ont observé le même phénomène par exemple, Kelley (1960) remarquait qu'en s'approchant de l'optimum, les plans de coupes successifs sont de plus en plus parallèles occasionnant ainsi des problèmes d'instabilité numérique.

Par conséquent, nous devons porter nos efforts sur la génération de nouvelles coupes "profondes" très distinctes de la coupe générée par la méthode usuelle d'accumulation des contraintes.

Une première approche, qui est due à Wolfe (1961), consiste à tenter d'améliorer les linéatisations successives de la méthode de plan de coupe de Kelley (1960).

Wolfe (1960) considère le problème

$$\text{Min } R(x)$$

sujet à

$$Ax = b \qquad (3.5.1)$$
$$x \geq 0$$

où

$A \in \mathcal{M}_{m\times n}$, $x, b \in R^n$, R est convexe, différentiable.

La méthode consiste à résoudre à l'itération k le problème linéaire

$$\text{Min } z$$

sujet à

$$z - (\nabla R(x^i), x) \geq v_i, \quad i = 1,\ldots,k \qquad (3.5.2)$$
$$Ax = b$$
$$x \geq 0$$

où

$$v_i = R(x^i) - (\nabla R(x^i), x), \quad i = 1,\ldots,k.$$

Soit x^{k+1} la solution optimale de ce problème, la contrainte suivante

$z - (\nabla R(x^{k+1}),x) \geq v_{k+1}$ est alors ajoutée.

Sachant que les vecteurs duaux associés aux contraintes de (3.5.2) sont respectivement:

$\gamma = (\gamma_1,\ldots,\gamma_k)$ et $\xi = (\xi_1,\ldots,\xi_m)$ et, puisqu'en résolvant (3.5.2), on obtient aussi la valeur des variables duales optimales, la technique de Wolfe consiste à ajouter en surplus la coupe suivante:

$$z - (\nabla R(x^{k+2}),x) \geq v_{k+2}$$

où

$x^{k+2} = \sum_{i=1}^{k} \bar{\gamma}_i x^i$, $(\bar{\gamma},\bar{\xi})$ étant le couple de vecteurs duaux optimaux.

Remarquons que $\sum_{i=1}^{k} \bar{\gamma}_i = 1$, $\bar{\gamma} \geq 0$ car $(\bar{\gamma},\bar{\xi})$ est une solution duale réalisable (optimale).

L'idée intuitive de Wolfe provient d'une approximation des conditions de Kuhn & Tucker du problème (3.5.1).

Nous avons appliqué cette technique d'accélération à notre méthode d'accumulation des contraintes:

ALGORITHME 3.5.1

Soit $x^1,\ldots,x^k \in \Omega$ des points réalisables connus, il s'agit d'abord de calculer la solution optimale u^k du problème Min-Max suivant:

$$\min_{u\in\Omega} \max_{i=1,2,\ldots,k} (C(x^i),u-x^i) \qquad (3.5.3)$$

puis, de calculer $y^k \in \Omega$ solution du problème

$$h(y^k) = \min_{\theta\in[0,1]} h(x^k+\theta(u^k-x^k))$$

où $h(x) = (C(x),x^k-u^k)^2$

comme précédemment; puis, à l'aide du vecteur dual optimal $(\bar{\gamma},\bar{\xi})$ de (3.5.3), nous posons

$$\tilde{y}^k = \sum_{i=1}^{k} \bar{\gamma}_i \, x^i$$

de sorte qu'à l'itération k+1, les deux coupes ci-contre sont générées:

$$(C(y^k), u-y^k) \quad \text{et} \quad (C(\tilde{y}^k), u-\tilde{y}^k).$$

La méthode présente certains désavantages. Elle nécessite le stockage des points $x^1,\ldots,x^k$ qui n'avait pas été nécessaire jusqu'à maintenant. Elle exige aussi le calcul d'une combinaison linéaire convexe de vecteurs causant des instabilités numériques souvent difficiles à éviter.

Lors des essais effectués sur un petit réseau (voir appendice A), nous avons remarqué que cette technique avait un rendement négatif, autant au niveau du nombre d'itérations que du temps de calcul. Nous avons remarqué à certains moments dans les expériences effectuées une tendance vers la génération de coupes presque parallèles, ce que nous n'avions pas rencontré dans la version originale de notre méthode d'accumulation des contraintes.

Nous nous rendons bien compte que la génération de coupes supplémentaires ne doit pas faire intervenir des calculs trop complexes et doit utiliser de l'information autre que celle de la méthode de coupe usuelle.

En se basant sur l'algorithme de Frank & Wolfe (1956) pour résoudre un problème d'équilibre avec des coûts "symétriques", nous proposons une nouvelle méthode pour générer une coupe supplémentaire. Soit x^k le point courant et u^k la solution optimale du problème linéaire dans la méthode d'accumulation des contraintes à l'étape k, alors pour déterminer le point qui formera cette deuxième coupe, nous devons calculer un plus court chemin associé aux coûts $C(x^k)$ noté u_{x^k} et résoudre le problème unidimensionnel

$$\theta^k = \arg\min_{\theta\in[0,1]} ((C(x^k+\theta(u_{x^k} - x^k)), x^k - u_{x^k}))^2.$$

Le vecteur $x^k+\theta^k(u_{x^k}-x^k)$, sera le point cherché.

Remarquons aussi que le point courant de l'étape suivante sera celui obtenu à partir des points x^k et u^k.

Malheureusement, les essais effectués sur le réseau de Hull nous amènent à la conclusion que la résolution d'un problème de plus court chemin pour chaque paire O/D et le calcul d'un deuxième pas afin de générer une coupe supplémentaire ne sont pas justifiés.

Par conséquent, nous reviendrons à la formulation de base de la méthode d'accumulation des contraintes et nous tenterons de simplifier le calcul du pas à chaque itération. En effet, la convergence de notre algorithme de coupes est assurée peu importe le choix du pas pourvu qu'il soit compris dans $(0,1]$.

Le premier choix qui nous vient à l'esprit est le pas prédéterminé $1/(k+1)$ où k est le numéro de l'itération courante; les résultats obtenus sont très mauvais. Nous avons considéré aussi le pas suivant suggéré par Karmanov (1977):

$$\theta_k = \text{Min } \{1, \rho_k x_o^k / \|x^k - u^k\|\}.$$

Les résultats obtenus sont relativement meilleurs que ceux de notre premier choix. Toutefois, ils demeurent impraticables.

Jusqu'à la fin de cette section, nous allons plutôt considérer différentes stratégies d'amélioration de la méthode d'accumulation des contraintes dans le cas spécifique où les fonctions de coût sont "symétriques".

La méthode des tangentes parallèles (PARTAN) que l'on peut concevoir comme une implantation particulière de la méthode des gradients conjugués, est décrite dans Luenberger (1973). Cette approche a été utilisée par Guélat et Florian (1983) pour améliorer la vitesse de convergence de l'algorithme de Frank & Wolfe dans la résolution du problème de Beckmann.

En se basant sur cette stratégie, nous procédons comme suit:

soit x^k le point courant et u^k une solution approximative du problème linéaire à cette étape telle que l'objectif est strictement positif, comme d'habitude il s'agit d'abord de calculer le point $\bar{x}^k$ de telle sorte que l'objectif du problème de Beckmann est minimisé dans la direction $(u^k - x^k)$. Puis, il s'agit de vérifier si $(x^{k-1} - \bar{x}^k)$ est une direction de descente de l'objectif de Beckmann au point $\bar{x}^k$. Si oui, nous effectuons de nouveau un calcul de pas de façon à minimiser l'objectif dans cette direction, le nouveau point obtenu sera x^{k+1}. Si non, $\bar{x}^k$ sera le point x^{k+1} en question.

Une deuxième alternative a été considérée. Elle consiste à vérifier plutôt si $(u^{k-1}-\bar{x}^k)$ est une direction de descente au point $\bar{x}^k$.

Malheureusement, nous n'avons pu tirer avantage de cette approche.

A la lumière des essais effectués, il semble que nous ayions atteint un plafond dans ce type d'approche du moins.

3.6 COMPARAISON AVEC D'AUTRES ALGORITHMES DEJA EXISTANTS

Nous comparerons notre méthode d'accumulation des contraintes avec différents algorithmes développés par le passé. Plusieurs d'entre eux exigent soit des hypothèses très restrictives ou des calculs relativement complexes. Nous considérons d'abord le réseau de Hull où les fonctions de coût sont "symétriques".

Nous allons d'abord examiner l'algorithme de S. Zuhovickii et al. (1969) qui s'énonce comme suit:

"soit $x^o,\ldots,x^k$, il s'agit de déterminer une solution optimale du problème linéaire

$$\text{Max } x_o$$

sujet à

$$(C(x^i), x^i - u) \geq x_o, \quad i = 1,\ldots,k \qquad (3.6.1)$$
$$u \in \Omega$$

puis, de poser $x^{k+1} = u^k$, $k \leftarrow k+1$."

Cette approche est identique à notre méthode de coupe sauf qu'elle ne fait pas intervenir le calcul des pas. La convergence est assurée grâce à des hypothèses de forte monotonie de la fonction de coût (Auslender, 1976).

Notons aussi que le problème (3.6.1) doit être résolu de façon exacte pour s'assurer la convergence de l'algorithme. Les contraintes non-actives peuvent être rejetées.

Les résultats obtenus sont rapportés dans le tableau (3.6.1). Nous voyons immédiatement que la convergence extrêmement lente de cet algorithme le rend impraticable. Les résultats sont non comparables avec ceux de (3.4.1) et ceux de (3.6.2) car, ici le pas est posé égale à 1 de sorte que nous tirons un avantage très limité de la détermination de la solution du problème linéaire

courant. Presqu'à chaque itération, la résolution du problème linéaire consiste uniquement à faire entrer une variable d'écart et résoudre au plus deux problèmes de plus courts chemins. La norme $\|x^k - x^{k-1}\|$ indique aussi une différence minime de la solution courante à la solution précédente. Par la suite, nous avons pensé à un mode d'alternances des coupes générées par notre méthode et celles générées par l'algorithme de Zuhovickii et al. Malheureusement, dans les essais effectués, lorsqu'une coupe de la forme (3.6.1) est générée, toutes les coupes subséquentes sont aussi des coupes de cette forme.

Nous avons pensé aussi à générer les coupes sous une forme qui se rapproche beaucoup plus du schéma habituel que l'on rencontre dans la littérature. Il s'agit de calculer:

$$x^{k+1} = x^k + \theta^k(u^k - x^k)$$

où

$$\theta^k = \arg\min_{\theta \in [0,1]} (C(x(\theta)), x(\theta) - x^k)$$

$$\text{où } x(\theta) = x^k + \theta(u^k - x^k).$$

Cette approche est véritablement meilleure que celle de Zuhovickii et al. mais elle peut être améliorée sensiblement.

Pour éviter le calcul du pas à chaque itération, nous allons maintenant considérer une suite $\{\alpha_k\}$ de pas fixée à priori. Nous obtenons alors:

"Soit $x^o, \ldots, x^k$ et u^k un plus court chemin associé à $C(x^k)$, alors

$$x^{k+1} = x^k + \alpha_k(u^k - x^k)$$

etc."

Nous supposons que $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\alpha_k \to 0$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.

Dans le tableau (3.6.2), nous avons examiné le cas où $\alpha_k \equiv 1/k$. La première itération de cette méthode est identique à la première itération de l'algorithme de Zuhovickii ce qui explique ce saut brusque de la fonction objectif à cette étape.

Itération k	Temps cumulé (sec.)	Gap courant du problème linéaire (x_o^k)	Gap standard GAP(x^k)	Nombre de coupes conservées	Norme $\|x^k - x^{k-1}\|$	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins résolus	Valeur de l'objectif du problème de Beckmann au point x^k
0	0,801	-	18 011,39	-	-	-	-	37 590,31
1	1,594	18 011,39	15 512 884,64	1	8 643,18	1	1	3 396 897,46
2	7,605	18 010,94	15 430 665,91	2	162,69	8	7	3 379 034,95
3	9,998	18 010,48	15 348 452,25	3	162,70	3	2	3 361 173,56
4	12,373	18 010,03	15 266 243,76	3	162,71	3	2	3 343 313,30
5	14,755	18 009,56	15 184 040,49	3	162,72	3	2	3 325 454,19
6	17,138	18 009,10	15 101 842,55	3	162,74	3	2	3 307 596,25
7	19,547	18 008,63	15 019 650,00	3	162,75	3	2	3 289 739,50
8	21,939	18 008,16	14 937 462,96	3	162,76	3	2	3 271 883,95
9	24,314	18 007,68	14 855 281,53	3	162,77	3	2	3 254 029,63
10	26,715	18 007,20	14 773 105,76	3	162,78	3	2	3 236 176,55
50	132,972	17 986,62	12 200 947,79	3	162,80	3	2	2 677 168,35
100	258,731	17 872,95	8 120 982,34	3	162,81	3	2	1 790 746,37
200	497,779	17 596,44	4 320 904,01	5	162,83	1	0	968 858,22
300	793,064	17 457,19	3 184 389,74	5	162,84	6	4	720 922,92

Tableau 3.6.1 Algorithme de Zuhovickii et al. appliqué au réseau de Hull (cas symétrique)

Cette heuristique donne de bons résultats, elle est meilleure que les 2 algorithmes précédents tant au point de vue de l'espace-mémoire exigé, du temps de calcul et de la complexité des programmes en jeu. Elle possède un seul inconvénient, sa lente convergence près de l'optimum.

Puisque le problème de Hull possède des fonctions de coûts "symétriques", il peut être résolu par plusieurs algorithmes efficaces (Leblanc et al., 1975; Nguyen, 1974,1976). Nous donnons ici les résultats obtenus en appliquant l'algorithme de Frank & Wolfe à notre problème (tableau 3.6.3).

Nous voyons ici que notre méthode de coupe se compare favorablement avec l'algorithme de Frank & Wolfe (graphique 3.6.1). L'avantage de notre méthode est de former un plateau plus rapidement et plus profondément.

Dans le graphique 3.6.2, nous obtenons un rendement sensiblement équivalent entre l'approche de PARTAN, celle de restriction et la méthode de coupes proposée ici.

Par conséquent, nous pouvons affirmer sans hésitations que notre algorithme se comporte de façon excellente, lorsque le problème est "symétrique" du moins.

Nous voulons maintenant reprendre l'algorithme de Lawphongpanich et Hearn où la résolution de chaque sous-problème sera plutôt faite par la méthode d'accumulation des contraintes (voir section 2.1).

Le schéma de l'algorithme est donné ci-après. Il diffère du schéma de calcul de Lawphongpanich et Hearn en ce qui a trait au test de précision lors de la résolution de chaque sous-problème. La convergence est encore assurée comme nous le verrons.

Variante de l'algorithme de S. Lawphongpanich et D.W. Hearn

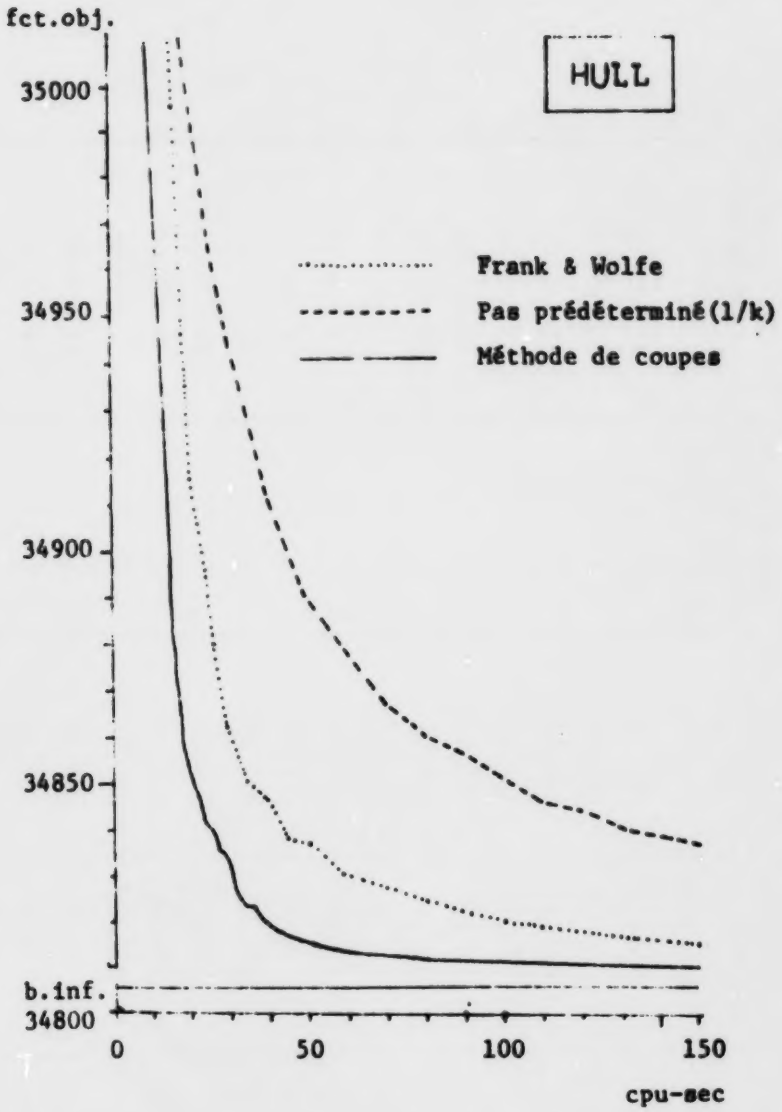
Etape 0. Soit $y^o \in \Omega$ un point extrême de Ω, posons $A^1 = \{y^o\}$, $\bar{G}^1 = -\infty$, $t=1$. Soit δ un facteur de tolérance et $\{\varepsilon_t\}$ une suite monotone où $\varepsilon_t > \varepsilon_{t+1} > 0$, $\varepsilon_t \to 0$.

Itération k	Temps cumulé (sec.)	Gap standard $GAP(x^k)$	Valeur de la norme $\|x^k-x^{k-1}\|$	Valeur de la fonction objectif du problème de Beckmann au point x^k.
0	0,793	18 011,39	+	37 590,31
1	1,505	643 615,30	4 321,59	174 858,12
2	2,269	100 146,46	2 006,88	57 478,28
3	3,029	30 199,95	1 172,27	42 014,37
4	3,779	11 101,01	626,06	37 875,41
5	4,531	4 890,47	385,11	36 522,64
6	5,279	2 708,61	189,35	35 960,72
7	6,026	1 749,20	173,06	35 678,73
8	6,771	1 282,43	171,70	35 519,49
9	7,514	994,55	138,36	35 412,46
10	8,256	838,39	138,36	35 342,34
15	11,979	439,92	98,58	35 173,53
20	15,706	339,49	71,75	35 083,75
30	23,157	210,31	45,30	34 991,73
40	30,622	157,65	24,66	34 946,07
100	75,677	72,56	20,12	34 863,76
199	150,087	32,10	7,76	34 836,08

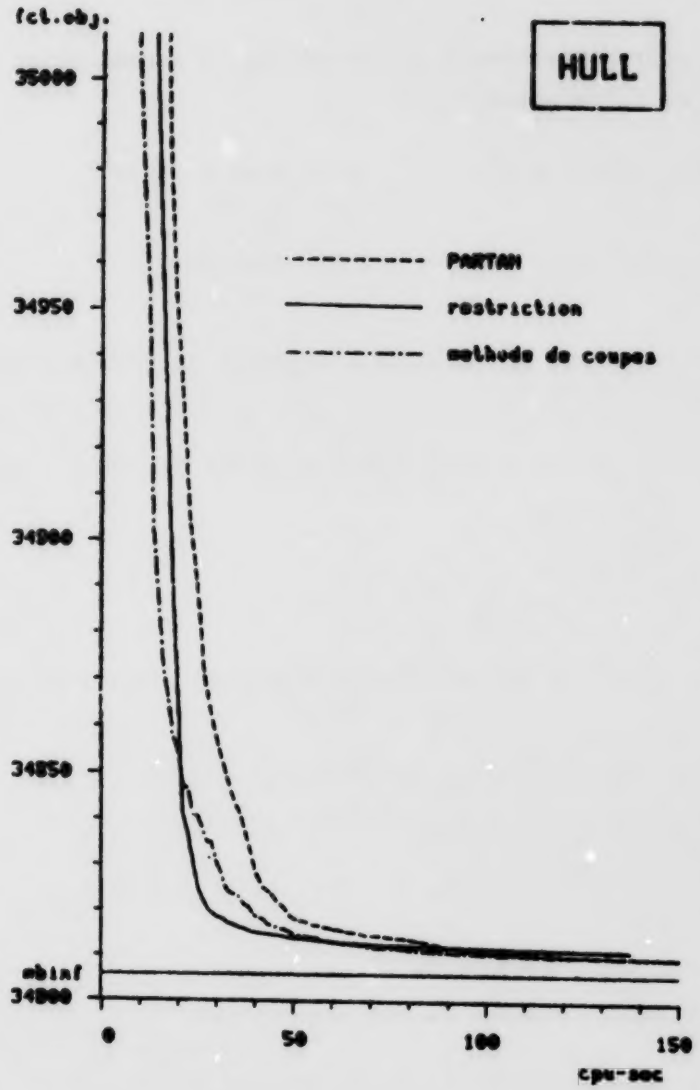
Tableau 3.6.2 Résultats obtenus avec le réseau de Hull en utilisant un pas prédéterminé (cas symétrique)

Itération k	Temps cumulé (sec.)	Gap courant GAP(x^k)	Valeur de la norme $\|x^k - x^{k-1}\|$	Nombre de problèmes de plus courts chemins	Valeur de la fonction objectif du problème de Beckmann au point x^k
0	0,786	18 011,39	-	-	37 590,31
1	2,127	7 537,19	1 041,21	1	36 287,98
2	3,380	1 823,36	964,84	1	35 545,07
3	4,639	1 538,92	465,37	1	35 374,53
4	5,743	903,60	253,22	1	35 313,06
5	6,858	994,16	368,95	1	35 246,81
6	7,972	613,05	166,93	1	35 219,96
7	9,223	650,74	212,68	1	35 176,41
8	10,482	611,82	142,44	1	35 149,43
9	11,722	694,23	422,21	1	35 071,97
10	12,975	488,67	164,34	1	35 048,16
15	18,545	261,61	109,45	1	34 946,80
20	24,237	222,61	165,36	1	34 893,08
30	35,521	99,93	89,16	1	34 850,41
40	46,686	63,18	36,75	1	34 837,93
100	111,077	19,92	16,31	1	34 817,90
199	211,238	11,07	11,34	1	34 812,65

Tableau 3.6.3 Résultats obtenus avec le réseau de Hull à partir de l'algorithme de Frank & Wolfe (cas symétrique)



Graphique 3.6.1: Comparaison entre l'algorithme de Frank & Wolfe, la méthode d'accumulation des contraintes et celle avec un pas fixe (réseau de Hull, cas symétrique)



Graphique 3.6.2: Comparaison entre l'approche de PARTAN, celle de restriction et la méthode d'accumulation des contraintes (Réseau de Hull, cas symétrique)

Etape 1. Résoudre approximativement par la méthode d'accumulation des contraintes la relation

$$x^t \in H(A^t),\ (C(x^t), x-x^t) \geq 0 \quad \text{pour tout } x \in H(A^t)$$

où

$H(A^t)$ = enveloppe convexe des colonnes de A^t.

$x_o^t \leq \varepsilon_t$,

x_o^t est la valeur optimale de l'objectif du problème linéaire à l'itération t.

Soit D^t = les colonnes de A^t ayant un poids nul dans l'expression de x^t.

Etape 2. Résoudre $(C(x^t), y^t) = \underset{y \in \Omega}{\text{Min}}\ (C(x^t), y)$.

Si $G(x^t) = (C(x^t), x^t - y^t)$ est égale à 0 alors c'est terminé.

Sinon

(I) si $G(x^t) \geq \overline{G}^t - \delta$ poser $A^{t+1} = A^t \cup \{y^t\}$

(II) si $G(x^t) < \overline{G}^t - \delta$ poser $A^{t+1} = (A^t \setminus D^t) \cup \{y^t\}$

poser $\overline{G}^{t+1} = \text{Min}\{\overline{G}^t, G(x^t)\}$, $t \leftarrow t+1$, aller à l'étape 1.

En se basant sur l'article de S. Lawphongpanich et D.W. Hearn. Nous examinons maintenant la convergence de cet algorithme.

Posons $\overline{T} = \{t: (C(x^t), x^t - z^t) < \varepsilon_t;\ t = 1,2,\ldots\}$,

où $(C(x^t), z^t) = \underset{z \in H(A^t)}{\text{Min}}\ (C(x^t), z)$.

LEMME 3.6.1: La suite $\{x^t\}_{t \in \overline{T}}$ est une sous-suite bien définie de $\{x^t\}_{t=1,2\ldots}$

DEMONSTRATION: Si $\{x^t\}$ est une suite finie, le lemme est vrai. Lorsque $\{x^t\}$ est une suite infinie, cela découle de la convergence des méthodes de coupe c'est-à-dire, qu'il existe une sous-suite qui converge vers le flot d'équilibre $\bar{x}^t$ du problème

$$\bar{x}^t \in H(A^t),\ (C(\bar{x}^t), x-\bar{x}^t) \geq 0 \quad \text{pour tout } x \in H(A^t),$$

lors de la résolution par les méthodes de coupe. □

Définissons maintenant $T = \{t \in \bar{T}: (C(x^t), x^t - y^t) < \bar{G}^t\}$, où y^t est un plus court chemin associé à $C(x^t)$. Nous pouvons vérifier facilement que les résultats de Lawphongpanich et Hearn sont encore vrais lorsqu'on considère cet ensemble T:

LEMME 3.6.2: La suite $\{x^t\}_{t \in T}$ est une sous-suite bien définie de $\{x^t\}_{t=1,2,\ldots}$

THEOREME 3.6.1:

La suite $\{x^t\}_{t \in T}$ converge à une solution d'équilibre du problème original.

L'avantage de cet algorithme est la résolution d'un seul problème de plus court chemin à chaque itération t, la simplification de la méthode d'accumulation des contraintes pour résoudre approximativement le probème d'équilibre à l'étape 1 de l'algorithme, et la rapidité à laquelle on peut déterminer une solution exacte de chaque sous-problème linéaire inhérente à la méthode de coupes.

En fixant $\delta = 0{,}1$ et en sachant que l'équilibre à l'étape 1 peut être calculé directement lorsque $\text{card}(H(A^t)) = 2$, nous avons regardé les 2 cas suivants (tableau 3.6.4 et 3.6.5):

a) $\varepsilon_t = 10.0/t$; b) $\varepsilon_t = 100.0/t$.

Les résultats obtenus sont similaires à ceux de la méthode de coupes lorsque la précision exigée dans le calcul de chaque sous-problème d'équilibre est assez grande. Par contre, nous remarquons que l'approche de Lawphongpanich et Hearn est lente au départ; cela prend plusieurs itérations avant de pouvoir exhiber une solution relativement bonne (voir graphique 3.6.3). En examinant les 2 tableaux (3.6.4) et (3.6.5), nous nous apercevons que le degré de précision ε_t n'a pas besoin d'être très fin pour des petites valeurs de t.

Malheureusement, par ce procédé, le nombre de points extrémaux conservés devient trop grand. Pour remédier à ces difficultés, nous allons considérer des valeurs de ε_t très petites pour réduire le nombre de points extrémaux à conserver. Dès lors, nous devons déterminer une solution approximative à chaque sous-problème linéaire rencontré dans la méthode de coupes afin de résoudre efficacement chaque sous-problème d'équilibre.

Par conséquent, nous devons nous servir du gap formé à partir du problème

$$\underset{x\in H(A^t)}{\text{Min}} \; (C(x^t),x)$$

pour mesurer le degré de précision de x^t; toutefois, le calcul supplémentaire de ce problème ne s'avère pas très avantageux (Tableau 3.6.6).

En effet, la méthode d'accumulation des contraintes est plutôt sensible au gap x_o^t mais, néanmoins, le comportement de l'algorithme de Lawphongpanich et Hearn et de ses variantes est très prometteur.

Pour compléter ce rapport, nous avons comparé notre approche avec celle de Frank et Wolfe, et de sa version améliorée (PARTAN) lors de la résolution du problème de Winnipeg (graphique 3.6.4).

Nous terminons cette section en considérant le problème "asymétrique", construit artificiellement à la section 3.4. Afin d'analyser le comportement de notre méthode de coupe et de pouvoir comparer celle-ci avec d'autres algorithmes existants, nous avons examiné les trois approches qui suivent.

En premier lieu, nous avons implanté la méthode de diagonalisation qui construit de façon récursive une suite $\{x^k\}$ où x^{k+1} est solution du problème d'optimisation de Beckmann formulé avec des coûts de transport définis de la façon suivante:

$$C_i^{k+1}(x) = C_i(x_1^k, x_2^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_M^k).$$

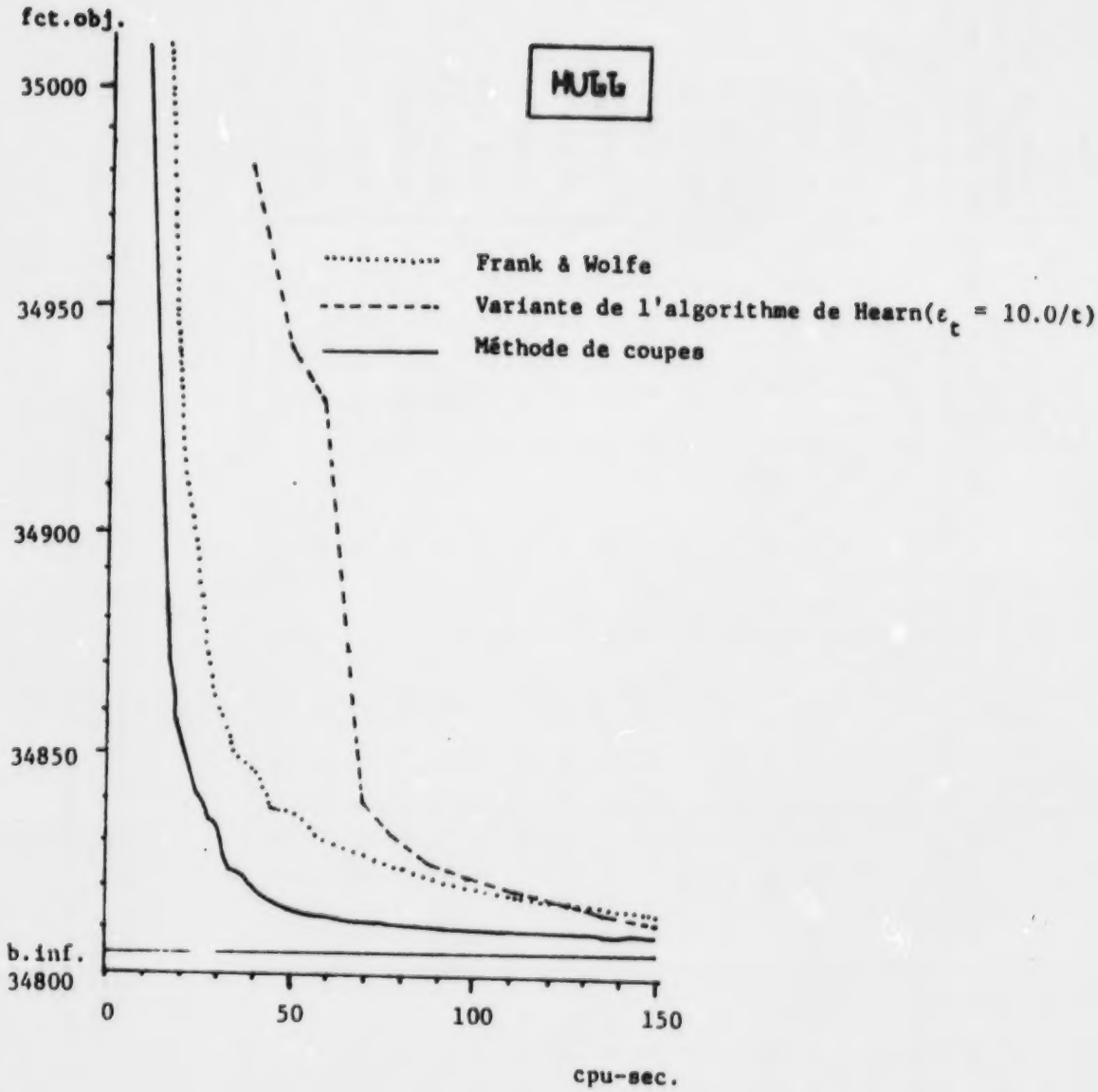
Pour résoudre le problème de Beckmann à chaque itération, nous avons utilisé notre méthode de coupes; α = 50,100 itérations de celle-ci ont servi à déterminer approximativement une solution d'équilibre au problème "symétrique" courant. Les résultats obtenus, indiqués aux tableaux 3.6.7 et 3.6.8 sont similaires. Par contre, ils sont légèrement supérieurs à ceux obtenus avec notre approche (voir tableau 3.4.5 et 3.4.6). La difficulté dans la méthode de diagonalisation est de fixer le degré de précision du point courant x^k.

Itération k	Temps cumulé (sec.)	Gap Standard (GAP(x^k))	Nombre de points extrémaux emmagasinés	Valeur de la norme $\|x^k - x^{k-1}\|$	Nombre de problèmes de programmation linéaire résolus	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins sur A^k résolus	Gap final (x_o^k)	Valeur de la fonction objectif du problème de Beckmann au point x^k
0	1,493	18 011,39	1	-	-	-	-	-	37 590,31
1	2,882	7 537,19	2	1 041,21	1	1	1	0,00	36 287,98
2	6,799	1 806,16	3	965,60	6	8	2	1,71	35 498,29
3	14,036	1 [illegible]95,13	4	620,35	10	12	3	3,26	35 249,89
4	18,375	774,47	5	188,63	6	16	6	0,90	35 216,90
5	29,058	891,28	6	793,68	15	18	4	1,59	35 030,08
6	39,541	502,59	7	288,45	14	23	7	0,88	34 981,65
7	50,694	363,93	7	332,52	15	25	7	0,18	34 940,91
8	57,078	356,57	8	134,29	9	18	6	0,24	34 929,62
9	70,244	129,15	8	384,22	16	25	9	1,04	34 839,30
10	77,260	130,14	9	138,15	9	17	5	0,54	34 831,39
15	134,460	42,82	11	46,44	10	28	13	0,62	34 815,66
20	200,915	12,17	13	40,28	14	55	25	0,47	34 810,30

Tableau 3.6.4: Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 10.0/t$) pour le réseau de Hull (cas symétrique)

Itération k	Temps cumulé (sec.)	Gap Standard ($GAP(x^k)$)	Nombre de points extrémaux emmagasinés	Valeur de la norme $\|x^k - x^{k-1}\|$	Nombre de problèmes de programmation linéaire résolus	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins sur A^k résolus	Gap final (x_o^k)	Valeur de la fonction objectif du problème de Beckmann au point x^k
0	1,499	18 011,39	1	-	-	-	-	-	37 590,31
1	2,885	7 537,19	2	1 041,21	1	1	1	0,00	36 287,98
2	6,429	1 804,93	3	966,35	5	6	2	14,07	35 498,30
3	11,965	1 342,53	4	611,08	7	9	3	16,71	35 257,42
4	16,079	744,18	5	160,55	5	15	7	6,14	35 219,20
5	23,089	534,85	6	875,98	9	11	3	11,85	34 966,45
6	26,962	386,94	7	437,99	5	8	4	11,00	34 913,57
7	32,098	307,10	8	178,50	6	14	6	10,66	34 896,44
8	36,956	305,37	9	160,73	6	9	4	11,62	34 877,67
9	44,310	154,78	9	298,99	8	19	10	10,95	34 843,35
10	48,452	131,25	10	145,46	5	8	4	9,30	34 835,54
15	76,562	55,42	15	38,27	5	16	9	4,67	34 820,82
20	106,151	27,99	19	33,18	6	13	7	4,97	34 815,67

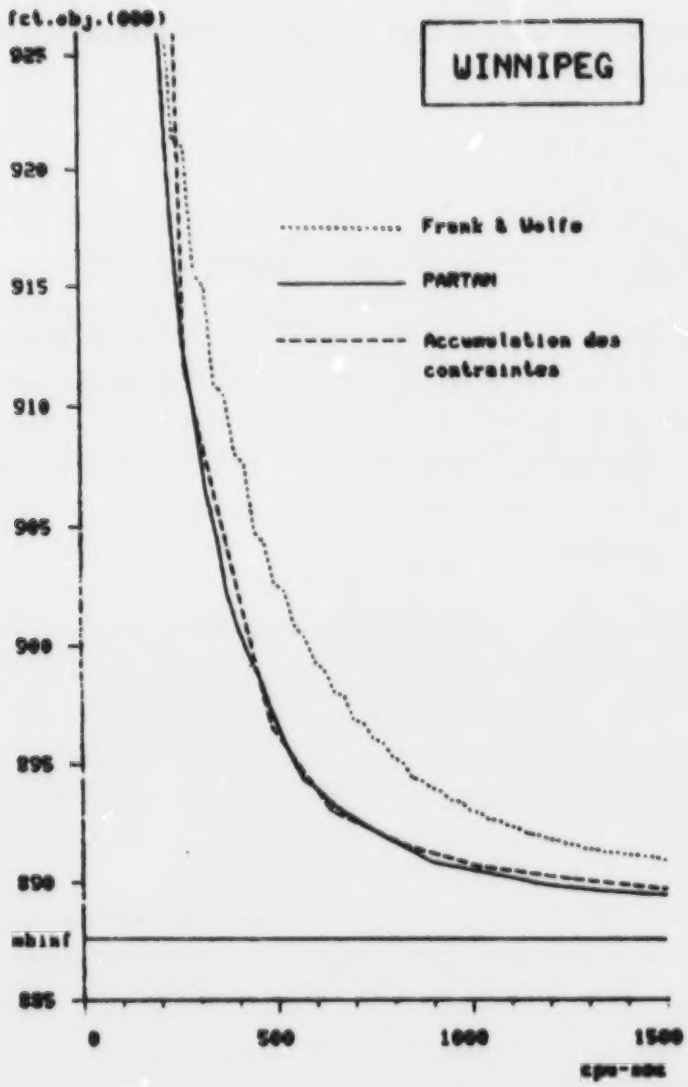
Tableau 3.6.5: Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 100.0/t$) pour le réseau de Hull (cas symétrique)



Graphique 3.6.3: Comparaison entre l'algorithme de Frank & Wolfe, la méthode d'accumulation des contraintes et une variante de l'algorithme de Hearn (cas symétrique)

Itération k	Temps cumulé (sec.)	Gap standard ($GAP(x^k)$)	Nombre de points extrémaux conservés	Valeur de la norme $\|x^k-x^{k-1}\|$	Nombre de problèmes de programmation linéaire résolus	Nombre total d'itérations de GUB	Nombre problèmes de plus courts chemins sur A^k résolus	Valeur de la fonction objectif du problème de Beckmann évaluée au point x^k
0	1,501	18 011,39	1	-	-	-	-	37 590,31
1	2,890	7 537,19	2	1 041,21	1	1	1	36 287,98
2	6,486	1 804,93	3	966,35	5	6	2	35 498,30
3	16,921	1 175,32	3	622,37	15	17	3	35 248,88
4	21,202	788,67	4	196,19	7	17	6	35 216,12
5	36,946	922,27	5	812,69	22	25	4	35 029,84
6	55,852	499,08	6	300,87	29	51	15	34 981,24
7	76,959	357,49	6	335,72	31	56	17	34 940,78
8	113,205	309,22	7	267,41	58	126	41	34 917,22
9	134,864	308,57	7	426,04	29	36	5	34 853,91
10	160,844	122,22	4	205,39	41	80	23	34 833,14
11	163,975	121,46	5	77,35	5	10	4	34 829,91
12	167,366	105,35	6	48,12	5	10	4	34 828,47
13	179,121	74,43	7	166,56	17	25	8	34 820,98
14	266,481	45,00	8	59,26	149	342	108	34 819,56

Tableau 3.6.6: Algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 10.0/t$) où chaque sous-problème d'équilibre est résolu par la méthode d'accumulation des contraintes (réseau de Hull, cas symétrique)



Graphique 3.6.4: Comparaison entre l'algorithme de Frank & Wolfe, l'approche de PARTAN et la méthode d'accumulation des contraintes (Réseau de Winnipeg, cas symétrique)

Considérons maintenant la méthode de projection définie à l'aide du processus itératif:

$$x^{k+1} = P_{\Omega}(x^{k}-\rho C(x^{k}))$$

où $x^{1} \epsilon \Omega$, ρ est convenablement choisi (strictement positif),
P_{Ω} est l'opérateur de projection sur Ω (Dafermos, 1980).

Etant donné que le calcul de x^{k+1} revient à déterminer le flot d'équilibre d'un problème de Beckmann, le point x^{k+1} a été obtenu après 50 itérations de la méthode d'accumulation des contraintes. De plus, après plusieurs essais sur ordinateur, ρ a été choisie égale à 300.0. Lorsque p s'écarte trop de cette valeur, la convergence s'avère très lente. Les résultats obtenus (tableau 3.6.9) sont nettement mauvais.

En dernier lieu, nous appliquons la variante précédente de l'algorithme de S. Lawphongpanich et D.W. Hearn où δ est fixé à 0.1 et ε_t = 100.0/t et 10.0/t respectivement. Les résultats obtenus (tableau 3.6.10 et 3.6.11) se comparent à ceux de l'algorithme 3.4.1 (tableau 3.4.6) et à la méthode de diagonalisation. Par contre, cette approche nécessite le stockage d'un nombre assez élevé de points extrémaux à certaines étapes.

Ce chapitre sur l'analyse des résultats, les techniques d'implantation et les stratégies pour accélérer la convergence de la méthode de coupe est loin d'être exhaustif. Plusieurs approches (comme par exemple, la décomposition de l'inverse de la base sous la forme LU, de nouveaux critères d'entrée des variables de base, etc.) suggérées par Benichou et al. (77), Ho et Loute (80) pour améliorer l'efficacité de l'algorithme de Dantzig-Wolfe pourraient être examinées. De même, des algorithmes plus performants pour trouver un plus court chemin sur un réseau peuvent être envisagés (Florian et al., 1979; S. Pallotino, 1979; etc.).

De façon à exploiter la structure particulière du sous-problème linéaire à résoudre à chaque itération de la méthode d'accumulation des contraintes, et aussi, arriver à le résoudre en un temps raisonnable, nous nous sommes inspirés de la forme révisée du simplexe et nous avons adapté la technique de décomposition de Dantzig-Wolfe (1963).

Itération k	Temps cumulé (sec.)	Gap standard ($GAP(x^k)$)	Nombre total d'itérations de GUB à l'étape k	Nombre total de problèmes de plus courts chemins résolus à l'étape k
0	0,768	21 448,50	-	-
1	60,901	40,71	118	44
2	117,589	38,60	124	46
3	168,799	36,86	116	43
4	222,919	26,46	126	45
5	273,599	6,39	119	41
6	325,811	17,20	117	45
7	377,218	7,80	115	45
8	427,655	8,57	120	43
9	480,018	19,36	119	46
10	532,044	6,22	121	45

Tableau 3.6.7: Résolution du problème "Asymétrique" à partir de la méthode de diagonalisation (la solution x^k avec coûts symétriques diagonalisés est obtenue après 50 itérations de la méthode d'accumulation des contraintes)

Itération k	Temps cumulé (sec.)	Gap standard ($GAP(x^k)$)	Nombre total d'itérations de GUB à l'étape k	Nombre total de problèmes de plus courts chemins résolus à l'étape k
0	0,770	21 448,50	-	-
1	117,916	51,19	243	90
2	226,185	23,52	242	86
3	329,306	24,71	238	88
4	433,391	30,60	228	89
5	535,694	14,15	232	86

Tableau 3.6.8: Résolution du problème "asymétrique" à partir de la méthode de diagonalisation (la solution x^k avec coûts symétriques diagonalisés est obtenue après 100 itérations de la méthode d'accumulation des contraintes)

Itération k	Temps cumulé (sec.)	Gap standard (GAP(x^k))	Nombre total d'itérations de GUB à l'étape k	Nombre total de problèmes de plus courts chemins résolus à l'étape k
0	0,825	21 448,50	-	-
1	37,014	3 758,32	118	42
2	72,632	2 234,25	124	42
3	108,161	2 563,49	124	43
4	146,570	1 764,29	131	47
5	185,497	3 961,89	126	48
10	365,425	1 535,94	118	41

Tableau 3.6.9: Résolution du problème "asymétrique" à partir d'une méthode de projection (la solution x^k avec coûts "symétriques" diagonalisés est obtenue après 50 itérations de la méthode d'accumulation des contraintes)

Itération k	Temps cumulé (sec.)	Gap standard ($GAP(x^k)$)	Nombre de points extrémaux emmagasinés	Valeur de la norme $\|x^k - x^{k-1}\|$	Nombre de problèmes de programmation linéaire résolus	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins sur A^k résolus	Gap final (x_o^k)
0	1,509	21 448,50	1	-	-	-	-	-
1	3,122	11 302,32	2	701,42	1	1	1	0,00
2	7,617	2 913,24	3	1 208,64	6	10	3	2,68
3	16,320	1 826,83	4	675,52	11	15	4	1,81
4	20,731	1 285,34	5	257,80	6	13	5	2,07
5	30,654	1 617,02	6	1 097,35	13	15	3	1,44
6	39,627	918,38	6	552,91	12	17	4	1,23
7	47,706	609,54	7	364,09	11	14	3	0,78
8	56,468	446,86	4	442,03	11	22	8	1,18
9	60,266	204,67	5	100,01	5	15	6	0,72
10	65,631	167,06	6	132,99	6	24	11	0,12
15	117,585	78,00	11	90,93	15	45	18	0,51
20	250,659	52,89	16	50,36	30	141	60	0,49

Tableau 3.6.10: Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 10.0/t$) (cas asymétrique)

Itération k	Temps cumulé (sec.)	Gap standard ($GAP(x^k)$)	Nombre de points extrémaux emmagasinés	Valeur de la norme $\|x^k - x^{k-1}\|$	Nombre de problèmes de programmation linéaire résolus	Nombre total d'itérations de GUB	Nombre de problèmes de plus courts chemins sur A^k résolus	Gap final (x_o^k)
0	1,511	21 048,50	1	-	-	-	-	-
1	3,137	11 302,32	2	701,42	1	1	1	0,00
2	7,137	2 911,00	3	1 210,33	5	8	3	14,69
3	13,585	1 860,71	4	719,05	8	12	4	18,52
4	17,321	1 290,75	5	267,27	5	8	3	10,09
5	23,685	1 567,28	6	998,59	8	10	3	19,14
6	30,909	942,77	7	525,79	9	14	5	11,83
7	36,595	702,07	8	332,28	7	9	3	7,09
8	42,441	437,79	8	441,41	6	18	9	11,67
9	49,408	322,76	9	184,78	8	16	7	3,58
10	56,963	191,30	10	180,08	9	14	5	7,59
15	84,338	49,33	15	49,11	6	17	10	6,40
20	116,089	31,80	19	16,31	3	6	4	4,16

Tableau 3.6.11: Variante de l'algorithme de Lawphongpanich et Hearn ($\varepsilon_t = 100.0/t$) (cas asymétrique).

La stratégie adoptée consiste à utiliser comme solution initiale du problème linéaire courant, la solution optimale du problème linéaire précédent. Malheureusement, nous avons dû introduire pour cela une variable artificielle donnant lieu à une forme composite de la fonction objectif. Ce procédé fait intervenir un paramètre ω qui doit être fixé au début de l'algorithme.

Mais, lorsque nous voulons résoudre exactement le problème linéaire à une étape donnée, cela exige souvent plusieurs itérations de la méthode de Dantzig-Wolfe. Nous pouvons alors nous demander si cette lenteur dans la résolution de chaque sous-problème linéaire dépendait surtout du procédé utilisé.

Comme la résolution du sous-problème linéaire est essentiellement une postoptimisation due à l'ajoût d'une contrainte additionnelle, l'idéal aurait été d'appliquer l'algorithme dual du simplexe. Mais, ceci est impossible car cela exige la connaissance de tous les flots extrémaux de l'ensemble des flots réalisables.

Or, pour contourner cette difficulté, nous avons résolu le dual du sous-problème linéaire à chaque itération en appliquant alternativement l'algorithme primal du simplexe, l'algorithme dual du simplexe et une approche de relaxation. De cette façon, une solution optimale du problème primal est disponible directement.

Les premiers essais effectués, n'ont pas été fructueux. La complexité des calculs entraîne des problèmes d'instabilité numérique difficiles à surmonter. Toutefois, une meilleure compréhension du problème dual mérite d'être obtenue.

CHAPITRE 4

LIEN AVEC LA PROGRAMMATION SEMI-INFINIE

4.1 INTRODUCTION

Dans ce chapitre, nous voulons faire le parallèle entre la programmation linéaire semi-infinie et la recherche d'un flot d'équilibre sur un réseau. Ceci nous permettra de développer une variante à notre méthode d'accumulation des contraintes.

Définissons le problème général de programmation semi-infinie sous la forme suivante:

$$\text{Min } F(x)$$

sujet à

$$f(x,y) \leq 0 \text{ pour tout } y \in Y \qquad (4.1.1)$$
$$x \in X$$

où $Y \subset R^m$ un ensemble compact,

$X \subset R^m$ un ensemble compact,

$F: X_o \to R$, $f: X_o \times Y \to R$ sont différentiables,

$X_o \supset X$ un ensemble ouvert,

$\text{card}(Y) = +\infty$.

Différentes méthodes pour calculer des solutions approximatives de (4.1.1) ont été examinées. Hettich (1978) suggère une approximation du problème (4.1.1) où l'ensemble Y est remplacé par un nombre fini de points de Y. En fait, une extension de l'algorithme de plan de coupe central de Elzinga-Moore (1975) pour la programmation convexe a été faite pour une classe de programmes semi-infinis par Gribik et Kortanek (1977), Gribik (1978). Malheureusement, leur approche présuppose qu'il existe un $\hat{x} \in X$ tel que $\hat{x}$ n'est pas une solution optimale de (4.1.1) et $f(\hat{x},y) < 0$ pour tout $y \in Y$. Or, nous verrons que cette condition n'est pas vérifiée dans les modèles d'équilibre.

Lorsque la fonction f(x,y) peut s'écrire sous la forme

$$g(y) - h(y)^t x \qquad (4.1.2)$$

où

g et h sont continues,

et lorsque F est convexe et différentiable, une approche basée sur la méthode de plan de coupe de Kelley (1960) a été examinée par Gustafson et Kortanek (1973). L'inconvénient majeur de cette méthode est qu'elle exige le calcul exact du maximum de $f(x^k,y)$ sur Y à l'itération k. Nous devons aussi rejeter ce schéma de calcul dans le cadre des modèles d'équilibre.

Honstede (1978) propose un algorithme qui nécessite la résolution d'une suite de problèmes semi-infinis linéaires, c'est-à-dire, où f est de la forme (4.1.2). La convergence n'est pas assurée et les calculs sont très complexes pour ce qui est de notre problème.

D'autres techniques (Watson (1975), Laurent et Carasso (1976), Gustafson (1981), etc.) ont été élaborées dans ce contexte mais ne sont pas véritablement applicables ici.

Dans la prochaine section, nous allons formuler le problème d'équilibre comme un programme linéaire semi-infini et suggérer une adaptation de l'algorithme de plan de coupe central de Gribik et Kortanek (1977), Gribik (1978).

4.2 FORMULATION ET RESOLUTION DU PROBLEME D'EQUILIBRE COMME UN PROGRAMME LINEAIRE SEMI-INFINI

Considérons de nouveau l'inégalité complémentaire

$$x^* \in \Omega, \ (C(x), x-x^*) \geq 0 \quad \text{pour tout } x \in \Omega$$

qui peut s'écrire encore

"Trouver $x^* \in \Omega$ solution de

Min 0

sujet à

$$(C(y),x)-(C(y),y) \leq 0 \text{ pour tout } y \in \Omega \qquad (4.2.1)$$
$$x \in \Omega".$$

En posant $g(y) = -(C(y),y)$ et $h(y) = -C(y)$ dans (4.1.2), on voit facilement que ce problème en est un de programmation linéaire semi-infinie. Remarquons aussi que l'objectif de (4.2.1) peut être remplacé par une fonction c quelconque $F(x)$.

En supposant la continuité de la fonction de coût $C(x)$, une adaptation de l'algorithme de plan de coupe central de Gribik et Kortanek (1977), Gribik (1978) s'énonce comme suit:

ALGORITHME 4.2.1:

Etape 0. Soit y^1 un point réalisable initial de Ω,
posons $k \leftarrow 1$.

Etape 1. Soit (x^k,σ_k) une solution optimale du problème de programmation linéaire
Max σ
sujet à
$$(C(y^i),y^i-x) \geq \|C(y^i)\|\sigma \ , \ i=1,2,\ldots,k \qquad (4.2.2)$$
$x\in\Omega$

Etape 2. Si $\sigma_k = 0$ alors c'est terminé.

Etape 3. Trouver $y^{k+1}\in\Omega$ telle que $(C(y^{k+1}),y^{k+1}-x^k) \leq 0$.
Poser $k \leftarrow k+1$, retourner à l'étape 1.

Cet algorithme se distingue de la méthode d'accumulation des contraintes décrite au chapitre 2 de 2 façons différentes: le problème de programmation linéaire (4.2.2) est tel que le coefficient de σ de la k ième coupe n'est plus égal à l'unité ; aussi, le choix du point y^{k+1} à l'étape 3 est très vaste.

Sous l'hypothèse de stricte monotonie de la fonction de coût, nous allons démontrer la convergence de cet algorithme. Nous adaptons la preuve de Gribik et Kortanek (1977), Gribik (1978) en tenant compte du fait que la condition suivante n'est pas satisfaite:

"Il existe un $\hat{x}\in\Omega$ tel que $\hat{x}$ n'est pas une solution d'équilibre et $(C(y),y-\hat{x}) > 0$ pour tout $y\in\Omega$".

LEMME 4.1

En supposant les fonctions de coût positives et strictement monotones sur Ω, si l'algorithme génère une séquence infinie de points $\{\sigma_k\}_1^\infty$, alors $\lim_{j\to\infty} \sigma_j = 0$.

DEMONSTRATION

Soit x* l'unique solution d'équilibre, alors (x*,0) est réalisable pour chaque problème de la forme de (4.2.2). D'où, $\sigma_k \geq 0$ pour tout k. De plus, nous voyons facilement que $\sigma_k \geq \sigma_{k+1}$ pour tout k. Par conséquent, $\lim_{k\to\infty} \sigma_k = \bar{\sigma} \geq 0$.

Supposons que $\bar{\sigma} > 0$.

Nous savons que

$$\forall k \geq j,\ (C(y_j), y_j - x_k) \geq \|C(y_j)\|\sigma_k \qquad (4.2.3)$$

et

$$\forall j,\ (C(y_j), y_j - x_{j-1}) \leq 0 \qquad (4.2.4)$$

Soustrayant (4.2.4) de (4.2.3), on obtient:

$$\forall k \geq j,\ (C(y_j), x_{j-1} - x_k) \geq \|C(y_j)\|\sigma_k$$

et, de par l'inégalité de Cauchy-Schwartz, on a

$$\|C(y_j)\| \cdot (\|x_{j-1} - x_k\| - \sigma_k) \geq 0.$$

Puisque, $\|C(y_j)\| \neq 0$ pour tout j, nous avons $\|x_{j-1} - x_k\| \geq \sigma_k \geq \bar{\sigma}$ pour tout $j \leq k$. Cela contredit le fait que $\{x_k\}_1^\infty$ possède des points limites, Ainsi, $\bar{\sigma} = 0$. □

Nous montrons maintenant que l'on détermine effectivement un flot d'équilibre.

PROPOSITION 4.2.1:

Sous les hypothèses du lemme 4.1, si l'algorithme termine après k itérations, y^k est le flot d'équilibre sinon il existe une sous-suite $\{y^k\}_{K_1}$ convergeant vers l'équilibre.

DEMONSTRATION

La preuve est similaire à celle du théorème (2.2.2.2).

Soit x* le flot d'équilibre, si l'algorithme arrête à l'itération k, alors

$$(C(y^i), y^i - x^k) \geq 0 \quad \text{pour tout } i \leq k,$$

et, $\sigma_k = 0$. Mais, cela implique alors l'existence d'un indice $j \leq k$ tel que

$$(C(y^j), y^j - x^*) = 0.$$

Dans le cas où la fonction de coût est strictement monotone, on obtient:

$$y^j = x^*.$$

Par hypothèse, $\sigma_i > 0$ pour tout $i < k$ de sorte que j est égale à k. Si l'algorithme construit une suite infinie de points $\{x^k, \sigma_k\}_1^\infty$, tous dans un compact, et telle que $\sigma_k > 0$. Il existe alors une sous-suite K telle que

$$(C(y^k), y^k - x^*) < \|C(y^k)\|\sigma, \quad k \in K.$$

Sachant que la limite de toute sous-suite $\{\sigma_k\}_{K_1}$, $K_1 \subset K$ convergente est nulle, la sous-suite $\{y^k\}_{K_1}$ converge vers x*. □

Gribik (1978) mentionne que dans l'article de Gochet et Smeers (1972), l'algorithme de plan de coupe central a été appliqué à la programmation semi-infinie et l'algorithme de Kelley (1960) à la programmation convexe. L'algorithme de Kelley (1960) exigeait environ 3 fois plus d'itérations. Ainsi, la recherche de la contrainte la plus violée ne semble pas nécessaire pour obtenir une bonne performance.

Nous avons appliqué l'algorithme (4.2.1) au problème de la recherche d'un flot d'équilibre sur le réseau de Hull ("symétrique"), tout en générant le même type de coupes que celles retrouvées dans la méthode d'accumulation des contraintes. Il semble qu'en considérant cette variante, nous réduisions les problèmes d'instabilité numérique lorsque chaque sous-problème linéaire est résolu exactement.

CHAPITRE 5

APPROXIMATION EXTERIEURE DE L'ENSEMBLE DES FLOTS D'EQUILIBRE

5.1 INTRODUCTION

Nous présentons ici une classe d'algorithmes qui constitue une synthèse des méthodes d'accumulation des contraintes et de pénalisation extérieure en optimisation, pour résoudre le problème d'équilibre de trafic. L'idée générale des méthodes étudiées est de remplacer le problème de la recherche d'un flot d'équilibre par une suite de problèmes plus simples. Nous opérons par approximation extérieure de l'ensemble des flots d'équilibre. La continuité et la monotonie de la fonction de coût sont les seules conditions exigées.

Cette classe d'algorithmes comprend entre autres la méthode d'accumulation des contraintes que nous avons étudiée précédemment.

Dans le cas général, sans la symétrie du Jacobien du vecteur de coût C(x), il existe plusieurs méthodes heuristiques pour l'obtention d'un flot d'équilibre (Fisk et Nguyen, 1980). Notons en particulier une méthode de diagonalisation dont la convergence peut être obtenue avec des hypothèses additionnelles sur C(x) (Dafermos, 1980; Florian, 1981). La méthode de diagonalisation construit de façon récursive une suite $\{x^k\}$ où x^{k+1} est la solution du problème de Beckmann formulé avec les coûts de transport suivants:

$$c_i^{k+1}(x) \equiv c_i(x_1^k, x_2^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_M^k).$$

De même, en se servant de l'inégalité complémentaire, nous sommes aussi amenés à construire une suite $\{x^k\}$ où x^k est une solution approchée d'un certain problème de programmation convexe.

5.2 DESCRIPTION D'UNE CLASSE D'ALGORITHMES POUR LE CALCUL D'UN POINT D'EQUILIBRE

Soit le problème général suivant:

"Trouver $\bar{x} \in X$ tel que $F(\bar{x}) = \min_{x \in X} F(x)$ (5.2.1)

où X est un sous-ensemble non-vide de R^M, F est une fonctionnelle définie sur R^M."

Plusieurs approches ont été considérées dans la littérature pour résoudre le problème (5.2.1). Parmi elles se trouvent les méthodes de coupe qui ont déjà été examinées précédemment dans le cadre des problèmes de la recherche d'un flot d'équilibre, et les méthodes de pénalités extérieures qui consistent à remplacer le problème (5.2.1) par un problème sans contrainte où un terme de pénalité est ajouté à la fonction F.

Martinet (1972) propose un procédé qui englobe ces deux classes de méthodes. Il s'agit de calculer à la k ième itération le point x^k comme suit:

$$F(x^k) + \varphi_k(x^k) = \min_{x \in R^M} (F(x) + \varphi_k(x))$$

où le terme de pénalité φ_k prend ses valeurs dans $[0,\infty]$ et est nulle sur X.

Remarquons que φ_k diffère d'une fonction de pénalité usuelle puisqu'elle peut prendre une valeur nulle en dehors de X.

Ce schéma de calcul exige qu'à chaque itération k, φ_{k+1} fournisse une meilleure approximation de la fonction indicatrice de X (une fonction nulle sur X et égale à $+\infty$ ailleurs) que φ_k.

En posant

$$X = \{x \in \Omega \mid (C(v), x-v) \leq 0 \text{ pour tout } v \in \Omega\}$$

représentant ainsi l'ensemble des points d'équilibre du réseau, et en imposant uniquement des hypothèses de continuité et de monotonie sur la fonction de coût C(x), le problème (5.2.1) consiste alors à déterminer un flot d'équilibre qui minimise F.

Nous pouvons adapter facilement l'approche de Martinet (1972) au problème d'équilibre. Nous posons $F(x) \equiv 0$; l'extension peut être faite sans difficultés mais cela a été omis dans le but d'abréger le texte.

ALGORITHME 5.2.1

Etape 0. Soit $x^0 \in \Omega$ un point réalisable initial, une suite de nombres non-négatifs $\{\eta_k\}$, $\{\varepsilon_k\}$ tels que $\lim_{k\to\infty} \eta_k = 0$, $\lim_{k\to\infty} \varepsilon_k = 0$. Poser $k = 0$.

Etape 1. Soit $v^k \in \Omega$ tel que $(C(v^k), x^k - v^k) > 0$, une solution ε_k-optimale de

$$\operatorname{Max}_{v\in\Omega} (C(v), x^k - v),$$

S'il n'existe pas de tel point $v^k \in \Omega$, terminer, le point x^k appartient à X.

Posons

$$\varphi_{k+1}(x) = \varphi_k(x) + L((C(v^k), x - v^k))$$

où

L: $R \to R^+$ est telle que pour toute suite $\{t_k\}$ de réels, on a:

$$\limsup_{k\to\infty} t_k \leq 0 \quad \text{si} \lim_{k\to\infty} L(t_k) = 0.$$

Etape 2. Calculer x^{k+1} une solution η_{k+1}-optimale de $\min_{x\in\Omega} \varphi_{k+1}(x)$

Poser $k \leftarrow k+1$.

Aller à l'étape 1.

Nous avons supposé implicitement que $\varphi_o(x) = \begin{cases} 0 & \text{si } x \in \Omega \\ \infty & \text{sinon} \end{cases}$

Martinet (1972) donne une preuve de convergence de l'algorithme 5.2.1 c'est-à-dire, toute valeur d'adhérence de la suite $\{x^k\}$ est solution d'équilibre, et

$$\lim_{k\to\infty} \varphi_k(x^k) = 0.$$

Il nous reste maintenant à expliciter davantage la fonctionnelle L et par le fait même, celle de φ. Nous examinerons dans ce qui suit différents choix de L et de φ.

1er choix: $L(u) = \begin{cases} 0 & \text{si } u \leq 0 \\ u^2 & \text{si } u > 0 \end{cases}$

Nous obtenons ainsi à l'étape 1 de l'algorithme

$$\varphi_{k+1}(x) = \sum_{i=0}^{k} (\operatorname{Max}\{0, (c(v^i), x - v^i)\})^2,$$

une fonction convexe différentiable sur Ω telle que

$$\nabla\varphi_{k+1}(x) = 2 \sum_{i=0}^{k} C(v^i) \cdot \max\{0,(C(v^i), x-v^i)\}.$$

Par conséquent, la méthode de Frank-Wolfe (1956) peut servir à résoudre de façon approchée le problème

$$\min_{x\in\Omega} \varphi_{k+1}(x) \qquad (5.2.2)$$

Etant donné que $\nabla\varphi_{k+1}(x)$ est monotone, continue et non-négative si C(v) l'est, alors le calcul de x^{k+1} nous ramène au cas simple de la recherche d'un flot d'équilibre lorsque les coûts de transport sont monotones et "symétriques".

2e choix: $L(u) = \begin{cases} 0 \text{ si } u \leq 0 \\ u \text{ si } u > 0 \end{cases}$

Cela nous donne $\varphi_{k+1}(x) = \sum_{i=0}^{k} \max\{0,(C(v^i),x-v^i)\}$ une fonction convexe mais non-différentiable partout sur Ω, où

$$\sum_{i=0,1,\ldots,k} C(v^i)$$

telle que

$$(C(v^i),x-v^i) > 0$$

est un sous-gradient de φ_k au point $x\in\Omega$.

Dans ce cas, des méthodes de sous-gradients peuvent être utilisées pour résoudre (5.2.2).

3e choix: $L(u) = \begin{cases} 0 \text{ si } u \leq 0 \\ \infty \text{ si } u > 0 \end{cases}$

On a $\varphi_k(x) = \max_{i=0,\ldots,k-1} (C(v^i),x-v^i)$.

Nous obtenons alors le problème de programmation linéaire que l'on retrouve dans la méthode d'accumulation des contraintes examinée antérieurement.

D'autres choix de L et φ sont possibles. Par exemple,

- $L(u) = \begin{cases} 0 \text{ si } u \leq 0 \\ \exp(u) - 1 \quad \text{autrement} \end{cases}$

$$\varphi_k(x) = \sum_{i=0}^{k-1} \max\{0,-1+\exp((C(v^i),x-v^i))\}$$

- $L(u) = \begin{cases} 0 \text{ si } u \leq 0 \\ e^{u^2} - 1 \quad \text{sinon} \end{cases}$

$$\varphi_k(x) = \sum_{i=0}^{k-1} \text{Max}\{0, -1+\exp((C(v^i), x-v^i)^2)\}$$

- etc.

Il serait intéressant aussi d'examiner différents choix de F (autres que la fonction nulle). En effet, en choisissant F(x) strictement ou fortement convexe sur Ω, cela peut donner un meilleur taux de convergence vers une solution d'équilibre.

Parmi les choix de L et de φ précédents, le premier choix possède un avantage important; en effet, étant donné que φ est différentiable sur Ω dans ce cas, nous pouvons considérer une méthode de directions réalisables pour résoudre (5.2.2). Puisque les méthodes de directions réalisables telles que Frank et Wolfe sont relativement simples quant à leur implantation et que l'accès aux points $C(v^i)$ est réduit au minimum, l'intérêt de ce premier choix n'est sûrement pas négligeable. Pourtant, dans les premiers essais effectués, la convergence expérimentale s'avère lente. Il est difficile de fixer le degré de précision lors du calcul de x^k et de v^k.

En pratique, nous avons calculé v^k comme suit:

$$v^k = x^k + \theta^k(u^k - x^k)$$

où $\theta^k = \arg \min_{\theta \in [0,1]} C(v(\theta))^t (v(\theta) - x^k)$.

$v(\theta) = x^k + \theta(u^k - x^k)$,

u^k est une solution extrémale de $\min \{(C(x^k), u) \mid u \in \Omega\}$.

Les coupes générées par la méthode d'accumulation des contraintes ont donné de meilleurs résultats. Un examen plus attentif de ces différentes alternatives est à envisager.

Smith (1983B) propose deux types de fonctions objectifs concernant le problème d'affectation d'équilibre "asymétrique":

$$\sum_i (\text{Max}\{0,(C(x),x-y^i)\})^2$$

où y^i est un point extrémal de Ω,

où encore,

$$\max_i \{(C(x),x-y^i)\}.$$

Celles-ci ont une forme "duale" de celles examinées dans ce chapitre. Smith (1983A) considère la fonction différentiable générale suivante:

$\sum_i F^2((C(x),x-y^i))$, où F est une fonction donnée.

et suggère une méthode de descente pour déterminer un flot d'équilibre. En effet, chacune de ces fonctions étant non-négative pour tout flot réalisable et nulle uniquement à x^*, une direction de descente a été considérée. Toutefois, les fonctions objectifs introduites par Smith (83b) ne sont pas convexes généralement.

CHAPITRE 6

APPROCHE GENERALE DE CALCUL D'UN FLOT D'EQUILIBRE DANS LE CADRE DE LA PROGRAMMATION CONVEXE

6.1 INTRODUCTION

Nous allons maintenant considéré une hypothèse légèrement plus restrictive que la monotonie de la fonction de coût pour résoudre un problème d'affectation. En effet, nous supposerons que la fonction $h_y(x) = (C(x),x-y)$ est convexe sur Ω, pour tout $y\in\Omega$ fixé. Cette hypothèse est justifiée par le fait que dans certaines applications, la fonction $C(x)$ n'est pas monotone alors que $h_y(x)$ semble être convexe (Fisk & Nguyen, 1982).

D'abord, Auslender (1972) vérifie la monotonie de l'opérateur suivant sous cette hypothèse de convexité:

$$T\begin{pmatrix}x\\z\end{pmatrix} = \begin{pmatrix}C(x) + J_c(x)^t(x-z)\\C(x)\end{pmatrix} \text{ sur } \Omega \times \Omega.$$

De plus, il démontre l'équivalence suivante:

PROPOSITION 6.1.1

$(x^*,y^*) \in \Omega\times\Omega$ est un point de selle de $(C(x),x-y)$ par rapport à $\Omega\times\Omega$ si et seulement si

$$T\begin{pmatrix}x^*\\y^*\end{pmatrix}^t \begin{pmatrix}x-x^*\\y-y^*\end{pmatrix} \geq 0 \qquad \forall(x,y)\in \Omega\times\Omega \ .$$

Ainsi, les développements mathématiques des chapitres antérieurs sont aussi applicables à l'opérateur T. Visiblement, il y a aucun intérêt à reprendre les approches précédentes dans ce nouveau cadre de travail. Notre objectif est plutôt de souligner certains schémas de calcul qui n'ont pas encore été rencontrés jusqu'à maintenant et qui dépendent de manière intrinsèque de la convexité de h_y.

6.2 UNE METHODE BASEE SUR LA REPRESENTATION BABYCENTRIQUE DES CONVEXES

La méthode proposée ici consiste à calculer successivement la solution d'un problème relativement simple appartenant à la théorie des jeux et de résoudre un problème unidimensionnel.

Pour développer l'algorithme qui suit, nous nous sommes basés sur une approche de Auslender (1972) laquelle possède l'inconvénient suivant: à chaque itération, nous devons résoudre le problème convexe suivant:

$$\min_{x\in\Omega} h_y(x)$$

Celui-ci étant beaucoup trop difficile à résoudre, nous avons ramené le calcul à celui d'un problème unidimensionnel produisant le schéma de calcul suivant:

ALGORITHME 6.2.1

A partir de $\binom{x^1_1}{y^1_1} \in \Omega\times\Omega$, on construit des suites $\{\binom{x^j_j}{y^j_j}\}, j = 1,\ldots,k$ dans $\Omega\times\Omega$, $\{\binom{\lambda^k_j}{\eta^k_j}\}$, $j=1,\ldots,k$ dans R^2, $k = 1,2,\ldots,$ $\{\binom{\bar{x}^k}{\bar{y}^k}\}$ dans $\Omega\times\Omega$ de la façon suivante:

1) $\binom{x^j_j}{y^j_j}$, $j = 1,2,\ldots,k \rightarrow \binom{\lambda^k_j}{\eta^k_j}$ comme suit:

On pose $\lambda \in R^k$, $\in R^k$,

$$\phi_k(\lambda,\eta) = \sum_{i=1}^{k}\sum_{j=1}^{k} \lambda_j\eta_i\,(C(x^j),x^j-y^i),$$

et

$$\Lambda_k = \{\lambda \mid \sum_{i=1}^{k} \lambda_i = 1,\ \lambda \geq 0\},$$

alors $\binom{\lambda^k}{\eta^k}$ est un point de selle de ϕ_k sur $\Lambda_k \times \Lambda_k$.

2) Posons $\bar{x}^k = \sum_{j=1}^{k} \lambda^k_j x^j_j$, $\bar{y}^k = \sum_{j=1}^{k} \eta^k_j y^j_j$.

3) Soit $x^{k+1} \in \Omega$ tel que $h_{\bar{y}^k}(x^{k+1}) \leq h_{\bar{y}^k}(\bar{x}^k)$ et, $h_{\bar{y}^k}(x^{k+1}) = \underset{\theta \in [0,1]}{\text{Min}}\ h_{\bar{y}^k}(\bar{x}^k + \theta(\bar{y}^k - \bar{x}^k))$;
soit $y^{k+1} \in \Omega$ une solution extrémale associée au coût $C(\bar{x}^k)$.

4) S'il existe des indices j_1 et $j_2 \in \{1,2,\ldots,k\}$ tels que

$$x^{k+1} = x^{j_1} \quad \text{et} \quad y^{k+1} = y^{j_2}.$$

alors $\bar{x}^k$ est un flot d'équilibre et l'algorithme s'arrête. Sinon, on retourne à l'étape 1 en posant $k \leftarrow k+1$.

L'originalité de cette approche réside dans la représentation barycentrique de l'ensemble Ω. En effet, chaque point appartenant à l'ensemble réalisable Ω peut s'exprimer comme une combinaison linéaire convexe des points extrémaux de Ω.

La preuve de convergence suivante est basée en grande partie sur les résultats dans Auslender (1972).

PROPOSITION 6.2.1

S'il existe des indices j_1 et $j_2 \in \{1,2,\ldots,k\}$ tels que $x^{k+1} = x^{j_1}$ et $y^{k+1} = y^{j_2}$, alors $\bar{x}^k$ est un flot d'équilibre et l'algorithme s'arrête. Sinon, toute valeur d'adhérence $(\bar{x},\bar{y})$ de la suite $\{(\bar{x}^k,\bar{y}^k)\}$ (et il en existe au moins une! puisque tout $(\bar{x}^k,\bar{y}^k)$ appartient à un compact) est telle que $\bar{x}$ est un flot d'équilibre.

DEMONSTRATION :

Puisque (λ^k) est un point de selle de ϕ_k sur $\Lambda_k \times \Lambda_k$, alors nous obtenons:
$\forall i,j \leq k$,

$$\sum_{i=1}^{k} \lambda_i^k (C(x^i), x^i - y^j) \leq \sum_{j=1}^{k} \eta_j^k (C(x^i), x^i - y^j) \qquad (6.2.1)$$

Comme $(C(x),x-y)$ est convexe-concave sur $\Omega\times\Omega$, la relation (6.2.1) entraîne que:

pour tout $i,j \leq k$, $(C(\bar{x}^k), \bar{x}^k - y^j) \leq (C(x^i), x^i - \bar{y}^k)$. (6.2.2)

a) Supposons qu'il existe des indices j_1 et $j_2 \in \{1,2,\ldots,k\}$ tels que

$$x^{k+1} = x^{j_1} \quad \text{et} \quad y^{k+1} = y^{j_2} .$$

La relation (6.2.2) entraîne alors:

$$(C(\bar{x}^k),\ \bar{x}^k - y^{k+1}) \leq (C(x^{k+1}),\ x^{k+1} - \bar{y}^k).$$

On sait de plus à partir de l'étape 3 de l'algorithme que

$$(C(x^{k+1}),\ x^{k+1} - \bar{y}^k) \leq (C(\bar{x}^k),\ \bar{x}^k - y^{k+1}).$$

D'où,

$$(C(x^{k+1}),\ x^{k+1} - \bar{y}^k) = (C(\bar{x}^k),\ \bar{x}^k - \bar{y}^k) = (C(\bar{x}^k),\ \bar{x}^k - y^{k+1}) = 0.$$

Mais, cela implique que $\bar{x}^k$ est un flot d'équilibre.

b) Soit $\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}$ une valeur d'adhérence arbitraire de la suite $\left\{\begin{pmatrix} \bar{x}^k \\ \bar{y}^k \end{pmatrix}\right\}$. Puisque Ω est convexe et compact, un tel point existe et on a évidemment $\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \in \Omega\times\Omega$. D'autre part, il existe un point $\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \in \Omega\times\Omega$ tel que $(\bar{x},\bar{y},\tilde{x},\tilde{y})$ soit une valeur d'adhérence de la suite $\{\bar{x}^k,\bar{y}^k,x^{k+1},y^{k+1}\}$ car la suite $\{x^{k+1},y^{k+1}\}$ est bornée; puisque la fonction de coût est continue, on obtient alors à partir des inégalités de l'étape 3 de l'algorithme par passage à la limite:

$$h_{\tilde{y}}(\bar{x}) \geq h_{\bar{y}}(\bar{x}) \quad \text{pour tout } y \in \Omega, \tag{6.2.3}$$

$$h_{\bar{y}}(\tilde{x}) \leq h_{\bar{y}}(\bar{x}) .$$

On obtient aussi après avoir fixé i,j dans (6.2.2) par passage à la limite:

$$h_{y_j}(\bar{x}) \leq h_{\bar{y}}(x^i) \text{ pour tout } i,j.$$

Et, en passant à la limite sur i,j, l'inégalité précédente devient:

$$h_{\tilde{y}}(\bar{x}) \leq h_{\bar{y}}(\tilde{x}).$$

En posant $y = \bar{y}$ dans (6.2.3), on obtient:

$$h_{\tilde{y}}(\bar{x}) \geq h_{\bar{y}}(\hat{x}).$$

Ainsi, on a:

$$h_{\tilde{y}}(\bar{x}) = h_{\bar{y}}(\tilde{x}) = h_{\bar{y}}(\bar{x}) \geq 0.$$

Si $h_{\bar{y}}(\bar{x}) > 0$ alors $h_{\bar{y}}(\hat{x}) < h_{\bar{y}}(\bar{x})$ d'après l'étape 3 de l'algorithme, ce qui est une contradiction.

Par conséquent, $h_{\bar{y}}(\bar{x}) = 0$ et $\bar{x}$ est une solution d'équilibre. □

Le calcul d'un point de selle de ϕ_k sur $X_k \times X_k$ se ramène à la résolution d'un jeu de stratégie auquel plusieurs méthodes efficaces ont été développées. De même, le calcul du pas à l'étape 3 consiste à résoudre un programme convexe en une dimension qui est très facile.

Enfin, l'espace-mémoire exigé se compare sensiblement à celui de la méthode d'accumulation des contraintes. Dans le prochain paragraphe, nous examinerons d'autres avenues possibles.

6.3 AVENUES POSSIBLES

L'objectif de ce chapitre n'étant pas d'énumérer de façon exhaustive toutes les approches de calcul possibles lorsque $(C(x),x-y)$ est convexe sur Ω pour tout $y \in \Omega$ fixé, mais plutôt de donner quelques orientations possibles.

Sachant que $\max_{y \in \Omega} C(x)^t(x-y)$ est convexe mais non-différentiable partout, une extension et une généralisation de la méthode de Kelley (1960) est proposée par Auslender (1972), Lemaire (1970). Des algorithmes "sans accumulation" sont aussi présentés lorsque la convexité est stricte (Martinet, 1972).

Une seconde approche (voir Rockafellar (76), Rockafellar (78)) consiste à ramener le problème de la recherche d'un flot d'équilibre à celui de déterminer un point z tel que $0 \in T(z)$, où T est une multi-application donnée. Cela découle d'un résultat de Minty (1964) qui indique que le sous-différentiel d'une

fonction convexe semi-continue inférieurement, non-identique à $+\infty$ est maximal monotone.

Enfin, plusieurs auteurs (Dem'Yanov, Malozemov, 1972; Fichevet, 1976; Butz, 1967; Dem'Yanov, 1966; etc.) ont abordé le problème de calculer un point de selle d'une fonction convexe-concave. Ils ont proposé des méthodes d'approximations successives pour résoudre leur problème. Celles-ci sont toutefois inapplicables directement au problème d'équilibre.

CHAPITRE VII

Méthodes de points fixes, de projection, de sous-gradients,...

7.1 Introduction

A partir de l'inégalité variationnelle

$$(C(x^*),\ x-x^*) \geq 0 \qquad \forall\ x \in \Omega$$

la recherche d'un flot d'équilibre se ramène soit à la résolution d'un problème MIN-MAX, soit à la résolution d'un problème MAX-MIN complémentaire, ou encore, au calcul d'un point de selle d'une fonction de la forme $(C(x),\ x-y)$ (voir chapitre I).

En considérant ces trois formulations d'un problème d'équilibre, plusieurs classes de méthodes dites de projection ou de points fixes sont discutées. L'exemple de Dafermos (1980) y est étudié dans ce cadre général. Cet exemple est nettement irréaliste à cause de la taille du réseau; mais, par contre, la solution de chaque sous-problème de projection est obtenue explicitement, facilitant ainsi l'analyse de ces algorithmes.

A la deuxième section, nous donnons un survol des méthodes dites de projection appliquées au problème d'équilibre. Les résultats proviennent surtout des articles de Sibony (1970), Dafermos (1980), Bertsekas et Gafni (1980).

Dans la section trois, nous adoptons l'approche générale de projection dans le cadre des trois formulations du chapitre I. Nous verrons

comment les propriétés inhérentes à ces formulations jouent un rôle important dans l'implantation et la convergence des algorithmes de projection.

On retrouve à la section quatre une généralisation primale-duale de l'algorithme de Dafermos (1980). A l'inverse de Dafermos (1980) qui calcule un point fixe solution de l'inégalité variationnelle de Smith (1979), nous cherchons plutôt un point de selle de la fonction $(C(x), x-y)$ sur $\Omega \times \Omega$.

Enfin, la généralisation obtenue à la section 4 est réexaminée en considérant les flots sur les chemins plutôt que les flots sur les arcs [Bertsekas et Gafni, 1980].

7.2 Survol des méthodes de projection pour résoudre un problème d'équilibre général

Dans cette section, nous considérons différents algorithmes de points fixes nous permettant de calculer des solutions d'équilibre. Sans hypothèse de symétrie du Jacobien de $C(x)$, Dafermos (1980) propose le schéma itératif suivant:

$$x^{k+1} = P_{\Omega}^{G}(x^{k} - \rho G^{-1} C(x^{k})), \qquad k=0,1,2,... \tag{7.2.1}$$

où la fonction de coût est supposée continuement différentiable et fortement monotone sur Ω, G est une matrice symétrique définie-positive et $\rho > 0$ est choisie à l'avance. P_{Ω}^{G} est l'opérateur de projection sur Ω au sens du produit scalaire $\langle x,y \rangle = (x,Gy)$.

Considérons la propriété suivante des opérateurs de projection [Kinderlehrer et Stampacchia, 1980] :

Proposition 7.2.1

Soit K un ensemble fermé et convexe sur un espace de Hilbert, alors $y = P_K(x)$, la projection de x sur K, si et seulement si

$$y \in K, \quad (y, \eta - y) \geq (x, \eta - y) \qquad \forall\, \eta \in K.$$

Le résultat suivant est alors obtenu.

Proposition 7.2.2

Chaque point fixe du processus itératif (7.2.1) est un flot d'équilibre.

Preuve :

$$\bar{x} = P_{\Omega}^{G}\left(\bar{x} - \rho G^{-1} C(\bar{x})\right)$$

$$\Longleftrightarrow$$

$$(\bar{x}, G(y - \bar{x})) \geq (\bar{x} - \rho G^{-1} C(\bar{x}),\ G(y - \bar{x})) \qquad \forall\, y \in \Omega \qquad \text{(prop. 7.2.1)}$$

$$\Longleftrightarrow$$

$$(C(\bar{x}),\ y - \bar{x}) \geq 0 \qquad \forall\, y \in \Omega.$$

Ce résultat est démontré dans Dafermos (1980). □

Proposition 7.2.3

Soit γ le maximum parmi les valeurs propres maximales des matrices symétriques définies positives

$$[J_C(x)]^t\, G^{-1}\, J_C(x), \qquad \forall\, x \in \Omega$$

alors, en prenant $0 < \rho < 2\alpha/\gamma$ où α est le paramètre de forte monotonie,

(7.2.1) possède un point fixe et $\|P_{\Omega}^{G}(x-\rho G^{-1}C(x)) - P_{\Omega}^{G}(y-\rho G^{-1}C(y))\|_G$ $\leq \lambda \|x-y\|_G \quad \forall x, y \in \Omega$, où $\lambda=[1-(1/\mu)\rho(2\alpha-\gamma\rho)]^{1/2}$, μ étant la valeur propre maximale de G, par rapport à la norme $\|x\|_G = (x, Gx)^{1/2}$. ("t" indique la transposée d'une matrice.)

Proposition 7.2.4

Soit x* l'unique flot d'équilibre, et $0< \rho< 2\alpha/\gamma$, $x^o \in \Omega$ quelconque, alors $x^k \to x^*$ lorsque $k \to \infty$,
et, $\|x^k - x^*\|_G \leq \lambda^k/(1-\lambda) \cdot \|x^1 - x^o\|_G$ où λ est définie à la proposition 7.2.3.

De plus, Dafermos (1980) nous fait remarquer que le processus itératif décrit en (7.2.1) nous amène à résoudre à l'itération k($k \geq 0$) un problème d'équilibre où la fonction de délai est affine, de la forme $Gx + (\rho C(x^k) - Gx^k)$ et le Jacobien de la fonction de délai est symétrique et défini positif.

Ultérieurement, Shu-Cherng Fang (1980) considère le schéma itératif (7.2.1) où il impose aux fonctions de coût la condition de Lipschitz et celle de forte monotonie. Il démontre la convergence lorsque G égale l'identité et ρ est un scalaire entre 0 et $2\alpha/L^2$, où $\alpha > 0$ est le paramètre de forte monotonie et $L > 0$ est la constante de Lipschitz. Dans le cas précédent où G est une matrice symétrique définie positive, ρ doit être compris entre 0 et $\frac{2\alpha\underline{u}^2}{uL^2}$, où u et $\underline{u}$ sont respectivement la plus grande et la plus petite valeur propre de G. Fang (1980) nous réfère aussi à plusieurs

méthodes de calcul de points fixes telles que celles de Merrill (1971), Eaves et Saigal (1972) et Todd (1976).

Remarquons toutefois qu'en (7.2.1), la projection est faite sur l'ensemble des flots sur les arcs (réalisables) ce qui peut causer des difficultés au niveau des calculs. Bertsekas et Gafni (1980) ont résolu ce problème en effectuant une transformation de variables de sorte que la projection est maintenant accomplie sur l'ensemble réalisable des flots sur les chemins. Ils proposent l'algorithme suivant:

$$y^{k+1} = P_{\Omega}^{G}(y^k - \rho G^{-1} A^t C(Ay^k)) \qquad k=0,1,2,... \qquad (7.2.2)$$

où y représente ici un vecteur de flots de chemins et A la matrice d'incidence arc-chemin du réseau G. La convergence de l'algorithme est démontrée lorsque ρ est choisie suffisamment petit et le vecteur de coût C est fortement monotone sur Ω, continu et satisfaisant la condition de Lipschitz:

$$\|C(x^1) - C(x^2)\|^2 \leq L\|x^1 - x^2\|^2 \quad \text{pour tous les flots d'arc réalisables } x^1 \text{ et } x^2. \qquad (7.2.3)$$

Puisque A^tCA n'est pas fortement monotone en général, la convergence n'est pas garantie par la théorie déjà existante [Sibony, 1970].

Remarquons que y^{k+1} est solution unique du problème multi-flot quadratique:

$$\underset{y\in\Omega}{\text{Min}} \ \frac{1}{2\rho}(y-y^k,\ G(y-y^k)) + (y-y^k,\ A^tC(Ay^k)) \qquad (7.2.4)$$

où Ω représente l'ensemble des flots de chemin réalisable.

La structure particulière des équations de conservation de flot

sur les chemins permet de décomposer (7.2.4) en autant de sous-problèmes indépendants que de paires O/D, chacun ne comportant qu'une seule contrainte linéaire. Cette approche facilite considérablement le calcul numérique de la projection, mais elle est très alourdie par le grand nombre de variables de flot de chemin, ce qui nécessite l'application d'une stratégie de restriction des variables.

Le taux de convergence est au moins linéaire. En effet, il existe un $\bar{\rho}>0$ tel que $\forall\ \rho\in(0,\bar{\rho}]$, la suite $\{y^k\}$ générée par (7.2.4) converge à une solution $y^*\in\Omega^*$ telle que

$$\|y^k-y^*\|_G \leq qB^k, \qquad k=0,1,2,\ldots$$

où $q>0$ et $B\in(0,1)$ dépendent de ρ et de y^o, $\Omega^*=\{y^*\in\Omega \mid x^*=Ay^*\}$, x^* étant l'unique flot d'arc à l'équilibre.

Jusqu'à maintenant, l'algorithme de Dafermos (1980) et celui de Bertsekas et Gafni (1980) ont été restreints à utiliser la même matrice G à chaque itération. Bertsekas et Gafni (1980) ont considéré une version plus générale de (7.2.2):

$$y^{k+1} = P_{\Omega}^{G_k}(y^k - \rho G_k^{-1}A^tC(Ay^k)), \quad k=0,1,2,\ldots \qquad (7.2.5)$$

où $\{G_k\}$ est une suite de matrices symétriques et définies positives où toutes les valeurs propres de chacune d'elles appartiennent à un intervalle compact des réels positifs. Les mêmes résultats de convergence sont obtenus lorsque la condition suivante est satisfaite:

$G_{k+1} = G_k$ si l'algorithme n'a pas progressé suffisamment,

c'est-à-dire en posant $w_k = (y^{k+1} - y^k,\ G_k(y^{k+1} - y^k))$,

$$\bar{w}_{k+1} = \begin{cases} \bar{B}w_k & \text{si } w_k \leq \bar{w}_k \\ \bar{w}_k & \text{si } w_k > \bar{w}_k \end{cases} \quad \text{où } \bar{B} \in (o,1),\ \bar{w}_o = \infty ,$$

$$G_{k+1} = G_k \quad \text{si } \bar{w}_{k+1} = \bar{w}_k .$$

Remarquons aussi que les conditions de convergence et les aspects numériques des méthodes de projection (7.2.1) et (7.2.2) ont été étudiés dans le cas général par Sibony (1970). Par exemple, il démontre la forte convergence de (7.2.1) lorsque ρ est choisi suffisamment petit dans le cas de fonctions de coûts fortement monotones satisfaisant une condition de Lipschitz. Il suggère de plus le schéma itératif (7.2.1) où ρ est variable; il pose $\rho = \rho_k = (C(x^k) - C(x^{k-1}),\ x^k - x^{k-1}) / \|G^{-1}(C(x^k) - C(x^{k-1}))\|_G^2$ dans le calcul de x^{k+1}. On voit facilement que $\rho_k \geq \alpha/\gamma$.

Différentes procédures mixtes sont aussi soulignées.

Malheureusement, ces méthodes de projection possèdent des inconvénients notables, à savoir le choix judicieux de la constante ρ et la résolution de chaque sous-problème de projection.

Dans la prochaine section, nous examinerons une approche de sous-gradient pour résoudre soit le problème MIN-MAX, soit le problème MAX-MIN complémentaire, soit le problème de la recherche d'un point de selle de $(C(x),\ x - y)$ sur $\Omega \times \Omega$.

7.3 Méthodes du sous-gradient

a) Résolution du problème MIN-MAX

Considérons en premier lieu la résolution du problème MIN-MAX suivant:

$$\text{Min} \quad f(x) \tag{7.3.1}$$

sujet à

$$x \in \Omega$$

où

$$f(x) = \underset{y\in\Omega}{\text{Max}} \ (C(x), x-y) .$$

Sachant que certaines sous-classes de fonctions de coûts (le cas affine, par exemple) sont telles que f(x) est convexe (strictement ou fortement convexe) sur Ω, nous allons examiner en détail cette situation.

Proposition 7.3.1

Si f(x) est convexe, $x^{k+1} = P_{\Omega}(x^k - \alpha_k (C(x^k) + J_C(x^k)^t (x^k - u^k)))$, $k=1,2,\ldots$

où $\alpha_k = \lambda_k / \|C(x^k) + J_C(x^k)^t (x^k - u^k)\|$, $\lambda_k \to 0$, $\sum_{k=1}^{\infty} \lambda_k = \infty$,

u_k est une solution extrémale associée à $C(x^k)$, $J_C(x^k)$ indique le Jacobien de C au point x^k, alors il existe une sous-suite de $\{x^k\}$ qui converge vers l'optimum de (7.3.1).

Preuve:

Polyak (1969) indique que la preuve de cette proposition est retrouvée dans

Polyak (1967). Ce dernier papier est disponible en russe seulement. Une preuve détaillée est donnée par Auslender (1976). □

Toutefois, cet algorithme possède une convergence très lente. Pour accélérer celle-ci, nous choisissons plutôt $\alpha_k = \lambda_k \cdot \frac{f(x^k)}{\|z^k\|^2}$ où $0 < \epsilon_1 \leq \lambda_k \leq 2 - \epsilon_2 > 0$, $z^k \in \partial f(x^k)$ ($\partial f(x)$ désigne le sous-différentiel de f au point x), $\epsilon_2 > 0$.

Proposition 7.3.2

Sachant que Ω est convexe et compact, qu'il existe au moins un optimum $x^* \in \Omega$, et que $f(x^*) = 0$, en supposant $f(x)$ convexe sur Ω et $\|z\| \leq d \ \forall z \in \partial f(x)$ où d est une constante donnée, alors $f(x^k) \to 0$ dans la méthode suivante:

$$x^{k+1} = P_\Omega(x^k - \alpha_k(C(x^k) + J_C(x^k)^t(x^k - u^k)))$$

$$\text{avec} \quad \alpha_k = \lambda_k \times \frac{(C(x^k), x^k - u^k)}{\|C(x^k) + J_C(x^k)^t(x^k - u^k)\|^2}, \quad 0 < \epsilon_1 \leq \lambda_k \leq 2 - \epsilon_2, \ \epsilon_2 > 0$$

tandis que x^k converge vers un point d'équilibre.

Preuve

Cette preuve est basée essentiellement sur le résultat de Polyak (1969). Sachant qu'une fonction continue et convexe possède une fonction de support à chaque point et que la projection sur un ensemble convexe et compact existe, alors la méthode énoncée à la proposition 7.3.2 est bien définie.

En utilisant la propriété suivante d'un opérateur de projection:

$\|P_\Omega(x) - P_\Omega(y)\| \le \|x - y\|$ $\forall\, x,y\in\Omega$, et en sachant que $f(x^*) \ge f(x^k) + (z^k, x^*-x^k)$, où $z^k \in \partial f(x^k)$, nous obtenons:

$$\begin{aligned}
\|x^{k+1} - x^*\|^2 &= \|P_\Omega(x^k - \alpha_k z^k) - x^*\|^2 \\
&\le \|x^k - \alpha_k z^k - x^*\|^2 \\
&= \|x^k - x^*\|^2 + \alpha_k^2\|z^k\|^2 - 2\alpha_k(z^k, x^k - x^*) \\
&\le \|x^k - x^*\|^2 + \frac{\lambda_k^2 (f(x^k))^2}{\|z^k\|^2} - 2\lambda_k \frac{f(x^k)}{\|z^k\|^2}\,(f(x^k)) \\
&\le \|x^k - x^*\|^2 - \frac{\varepsilon_1\varepsilon_2 (f(x^k))^2}{\|z^k\|^2}\,. \qquad (7.3.2)
\end{aligned}$$

D'où, $\|x^k - x^*\|$ est décroissant et $\|x^k - x^*\| \le \|x^0 - x^*\|$. Puisque $\|z^k\|^2$ est bornée pour tout k, alors $f(x^k) \to 0$. Ainsi, il existe une sous-suite de $\{x^k\}$ convergeant vers un point d'équilibre $\bar{x}$.

Nous montrons maintenant que la suite entière $\{x^k\}$ converge vers $\bar{x}$. Supposons qu'il existe deux suites, $x^{k_i} \to \bar{x}$ et $x^{\ell_i} \to \bar{\bar{x}}$, $\bar{x} \ne \bar{\bar{x}}$. Nous savons que $\|x^{k_i} - \bar{x}\| \to \rho_1$, $\|x^{\ell_i} - \bar{\bar{x}}\| \to \rho_2$. Sachant que $\bar{x}$ et $\bar{\bar{x}}$ sont des points d'équilibre et à partir de l'inégalité (7.3.2) où x^* est remplacé par $\bar{x}$ et $\bar{\bar{x}}$ respectivement, nous avons:

$\|x^k - \bar{x}\| \to \rho_1$, $\|x^k - \bar{\bar{x}}\| \to \rho_2$ et en particulier, $\|x^{\ell_i} - \bar{x}\| \to \rho_1$, $\|x^{k_i} - \bar{\bar{x}} \to \rho_2$.

Supposons que $\rho_1 \le \rho_2$. Alors, $\|x^{\ell_i} - \bar{x}\|^2 = \|x^{\ell_i} - \bar{\bar{x}} + \bar{\bar{x}} - \bar{x}\|^2 = \|x^{\ell_i} - \bar{\bar{x}}\|^2 + \|\bar{\bar{x}} - \bar{x}\|^2$

$$+ 2(x^{\ell_i} - \bar{\bar{x}}, \bar{\bar{x}} - \bar{x})$$

$$\rightarrow \rho_2^2 + \|\overline{\overline{x}} - \overline{x}\|^2 > \rho_1^2 \quad \text{ce qui est impossible}$$

puisque $\|x^{\ell_i} - \overline{x}\| \rightarrow \rho_1$.

□

Une preuve détaillée est aussi donnée par Auslender (1976).

Examinons maintenant le taux de convergence de cet algorithme.

En posant $\overline{\Omega} = \{x^* \in \Omega \mid x^* \text{ est un point d'équilibre}\}$, nous allons considérer la condition (7.3.3) suivante:

$$f(x) \geq m \operatorname{Min}_{y \in \overline{\Omega}} \|x-y\|, \ \forall x \in \Omega : \|x - P_{\overline{\Omega}}(x^o)\| \leq \operatorname{Min}_{y \in \overline{\Omega}} \|x^o - y\| \qquad (7.3.3)$$

où $m > 0$.

Polyak (1969) démontre alors la convergence de la méthode précédente au taux d'une série géométrique.

Proposition 7.3.3

Si f(x) satisfait la condition (7.3.3) et que les conditions de la proposition 7.3.2 sont vérifiées, alors cette dernière méthode converge à un point d'équilibre x* au taux d'une série géométrique:

$$\|x^k - x^*\| \leq \left[\left(\operatorname{Min}_{y \in \overline{\Omega}} \|x^o - y\| \right) \cdot \left(1 + \frac{2}{1-q} \right) \right] q^k$$

où $q = (1 - \varepsilon_1 \varepsilon_2 \frac{m^2}{c^2})^{1/2}$.

Preuve:

La démonstration de cette proposition est faite par Polyak (1969). Nous la

reprenons ici afin d'être complets.

A partir de la démonstration de la proposition 7.3.2, on trouve que

$$\|x^k - P_{\overline{\Omega}}(x^o)\| \leq \underset{y\in\overline{\Omega}}{\text{Min}} \|x^o - y\| \qquad \text{pour tout } k \ .$$

d'où

$$(\underset{y\in\overline{\Omega}}{\text{Min}} \|x^{k+1} - y\|)^2 \leq \|x^{k+1} - P_{\overline{\Omega}}(x^k)\|^2$$

$$\leq \|x^k - P_{\overline{\Omega}}(x^k)\|^2 - \frac{\varepsilon_1 \varepsilon_2 (f(x^k))^2}{\|z^k\|^2}, \quad z^k \in \partial f(x^k)$$

$$\leq (\underset{y\in\overline{\Omega}}{\text{Min}} \|x^k - y\|)^2 - \frac{\varepsilon_1 \varepsilon_2 m^2}{c^2} (\underset{y\in\overline{\Omega}}{\text{Min}} \|x^k - y\|)^2$$

$$= q^2 \cdot (\underset{y\in\overline{\Omega}}{\text{Min}} \|x^k - y\|)^2 \ .$$

Par conséquent, $\underset{y\in\overline{\Omega}}{\text{Min}} \|x^k - y\| \leq q^k \cdot \underset{y\in\overline{\Omega}}{\text{Min}} \|x^o - y\|$.

A partir des propriétés des opérateurs de projections, nous avons:

$$\|P_{\overline{\Omega}}(x^{k+1}) - P_{\overline{\Omega}}(x^k)\| \leq \|x^{k+1} - x^k\|$$

$$\leq \|x^{k+1} - P_{\overline{\Omega}}(x^k)\| + \|x^k - P_{\overline{\Omega}}(x^k)\|$$

$$\leq 2\|x^k - P_{\overline{\Omega}}(x^k)\|$$

$$\leq 2q^k \cdot \underset{y\in\overline{\Omega}}{\text{Min}} \|x^o - y\| \ .$$

D'où la suite $P_{\underline{\Omega}}(x^k)$ converge à un point d'équilibre x* avec

$$\|P_{\underline{\Omega}}(x^k) - x^*\| \leq \frac{2q^k}{1-q} \cdot \underset{y\in\underline{\Omega}}{\text{Min}} \|x^o - y\| \ .$$

Enfin, on a:

$$\|x^k - x^*\| \leq \|x^k - P_{\underline{\Omega}}(x^k)\| + \|P_{\underline{\Omega}}(x^k) - x^*\| \leq (q^k \cdot \underset{y\in\underline{\Omega}}{\text{Min}} \|x^o - y\|) \cdot (1 + \frac{2}{1-q}) = Kq^k \ .$$

□

Nous allons maintenant considérer un cas où la condition (7.3.3) est satisfaite.

Proposition 7.3.4

Si $\phi(x)$ est fortement convexe sur un ensemble X convexe et fermé où ρ est le paramètre de convexité forte, alors $\forall\, x \in X$, on a l'inégalité

$$\phi(x) - \phi(x^*) \geq \frac{\rho}{2} \quad \|x - x^*\|^2 \quad \text{où } x^* \text{ est le point minimum de}$$

$\phi(x)$ sur X.

Preuve: (Voir Karmanov (1977), p. 42-43).

En posant $\phi(x) = f^2(x)$ fortement convexe sur Ω, on obtient alors $f^2(x) \geq \frac{\rho}{2} \|x - x^*\|^2$ ce qui implique $f(x) \geq \sqrt{\frac{\rho}{2}} \cdot \|x - x^*\|$. Dans ce cas, $m = \sqrt{\frac{\rho}{2}}$.

Certes, les hypothèses de convexité sur f(x) sont plutôt restrictives, mais la sous-classe des fonctions de coût affines satisfait ces hypothèses et elle est souvent rencontrée en pratique. L'amélioration de la convergence de la méthode élaborée à la proposition 7.3.2 est rendue possible grâce à

remarque suivante de Polyak (1969):

en posant $Q_k = \{x \in \Omega \mid f(x^k) + (z^k, x - x^k) \leq 0,\ k \in I_k\}$, $x^{k+1} = P_\Omega P_{Q_k}(x^k)$ (7.3.4)

$$\lambda_k \equiv 1,\ I_k = \{k\},\ z^k \in \partial f(x^k),$$

on obtient la méthode précédente (proposition 7.3.2).

Maintenant, nous pouvons considérer I_k comme un sous-ensemble d'indices de $\{0,1,\ldots,k\}$ contenant k.

Lorsque nous prenons $I_k = \{0,1,\ldots,k\}$, l'approche obtenue a plusieurs traits en commun avec la méthode de plan de coupe de Kelley (1960). Toutefois, celle-ci ne possède pas le taux de convergence d'une série géométrique.

Polyak (1969) nous donne de façon explicite le cas $I_k = \{k-1,k\}$:

$$x^{k+1} = P_Q(x^k - \alpha_k z^k - \beta_k z^{k-1}),$$

avec $z^k \in \partial f(x^k),\quad z^{k-1} \in \partial f(x^{k-1}),$

où $\alpha_k = \dfrac{f(x^k)}{\|z^k\|^2}$ et $\beta_k = 0$ si $f(x^{k-1}) + (z^{k-1}, x^k - \dfrac{f(x^k) z^k}{\|z^k\|^2} - x^{k-1}) \leq 0$

sinon $\alpha_k = \{f(x^k)\|z^{k-1}\|^2 - (z^k, z^{k-1})(f(x^{k-1}) + (z^{k-1}, x^k - x^{k-1}))\}\ .$

$$\{\|z^k\|^2 \cdot \|z^{k-1}\|^2 - (z^k, z^{k-1})^2\}^{-1}$$

$$\beta_k = \{\|z^k\|^2 (f(x^{k-1}) + (z^{k-1}, x^k - x^{k-1})) - f(x^k)(z^k, z^{k-1})\}.$$

$$\{\|z^k\|^2 \|z^{k-1}\|^2 - (z^k, z^{k-1})^2\}^{-1}\ .$$

Mentionnons maintenant que ces derniers schémas itératifs font intervenir le Jacobien de la fonction de coût ce qui peut devenir un inconvénient majeur.

Les deux approches précédentes (propositions 7.3.1 et 7.3.2) ont été abordées par plusieurs autres auteurs. Par exemple, Shepilov (1976) considère la première approche où le pas est fixé a priori à chaque itération; il définit explicitement l'ensemble des points réalisables à partir de contraintes d'inégalités et s'assure la convergence de la méthode sans imposer la condition de Slater et l'existence d'un ensemble réalisable borné.

Polyak suggère un pas fixe $\alpha_k = \alpha > 0$ qui est réduit de moitié à chaque fois que $f(x^k)$ oscille autour d'une limite. De même, Shor (1968) examine le cas $\alpha_k = \alpha_0 q^k$, $0 < q < 1$. Mais ces deux méthodes sont difficilement implantables.

Shor (1968) tente aussi d'accélérer la convergence de la méthode du sous-gradient en considérant un opérateur de projection qui est transformé à chaque itération. Il examine deux versions qui nécessitent le stockage d'une matrice mise à jour au début de chaque itération. Malheureusement, l'espace-mémoire exigé est beaucoup trop exhorbitant dans le cas de problèmes de flots d'équilibre.

Dans le cas où le flot d'équilibre n'est pas unique, ces méthodes de projection ne sont pas stables; en effet, la convergence vers un flot d'équilibre x^* dépend du flot initial de départ. Polyak introduit alors une fonction de régularisation $f_1(x)$ strictement convexe sur Ω et considère le processus itératif $x_{k+1} = P_\Omega(x_k - \alpha_k(z^k + \varepsilon_k z_1^k))$ où $\varepsilon_k \to 0$, $z^k \in \partial f(x^k)$, $z_1^k \in \partial f_1(x^k)$. Ainsi, nous obtenons la stabilité, i.e. le flot d'équilibre x^* minimise $f_1(x)$. La proposition suivante a été démontrée par Bakushinskii et Polyak (1974).

Proposition 7.3.5

Soit $f(x)$ et $f_1(x)$ continues et convexes, $f_1(x)$ fortement convexe,

$$\varepsilon_k \to 0,\ \alpha_k/\varepsilon_k \to 0,\ \sum_k \varepsilon_k \alpha_k = \infty,\ \varepsilon_k/\varepsilon_{k+1} = 1 + 0\ (\alpha_k \varepsilon_k)$$

(par exemple, $\alpha_k = k^{-\frac{1}{2}}$, $\varepsilon_k = k^{-\rho}$, $0 < \rho < \frac{1}{2}$).

Alors, $x_k \to x^*$, un flot d'équilibre qui minimise $f_1(x)$.

Dans le cas de fonctions de coûts monotones et affines, cet algorithme nous permet de calculer un point d'équilibre bien spécifique.

Enfin, nous remarquons que le problème (7.3.1) peut s'écrire aussi:

$$\min_{x \in \Omega} \max_{1 \le j \le J} (C(x),\ x - y^j) \quad \text{où } y^j \text{ est un point extrémal de } \Omega,\ |J| < \infty .$$

D'où

$$\partial f(x) = \Big[\sum_{j \in I(x)} \lambda_j \nabla (C(x),\ x - y^j) \,\Big|\, \lambda_j \ge 0,$$

$$\sum_{j \in I(x)} \lambda_j = 1,\ I(x) = \{j:\ y^j \text{ est une solution extrémale associée aux coûts } C(x)\}\Big] .$$

Dans les algorithmes précédents, un point quelconque de $\partial f(x)$ peut être choisi.

b) Résolution du problème MAX-MIN convexe

Il s'agit de résoudre le problème de programmation convexe suivant:

$$\min_{x \in \Omega} g(x) \quad \text{où} \quad g(x) = \max_{y \in \Omega} (C(y),\ y - y) \quad \text{est une fonction convexe sur } \Omega .$$

Les méthodes de sous-gradient exactes [Auslender, 1976] ne sont pas applicables ici car le calcul de g à un point $x \in \Omega$ donné est impossible de façon exacte; par contre, on peut appliquer une méthode de projection en présence de perturbations [Polyak, 1971] comme suit:

$$x^{k+1} = P_{\Omega}(x^k - \alpha_k C(\tilde{y}^k))$$

où $\tilde{y}^k$ est une approximation de y^k solution optimale de $\underset{y \in \Omega}{\text{Max}}\ (C(y), x^k - y)$ et satisfaisant le critère $\|C(\tilde{y}^k) - C(y^k)\| \leq d\|x^{k+1} - x^k\|$.

En supposant g(x) différentiable sur Ω, g'(x) satisfaisant une condition de Lipschitz de constante M, et $0 < \varepsilon_1 \leq \alpha_k \leq 2/(M+2d) - \varepsilon_2, \varepsilon_2 > 0$, il montre que $\{g(x_k)\}$ est monotone décroissante et $\lim_{k \to \infty} g(x^k) = 0$, $g(x^k) = 0(1/k)$. Les conditions sur g(x) sont en général difficilement vérifiables; toutefois, on sait que g(x) est différentiable sur Ω lorsque C(x) est affine. D'autres critères d'approximation peuvent être utilisés et d'autres façons de choisir le pas ne faisant pas intervenir la constante M ont été étudiées. Pourtant, le calcul de $\tilde{y}^k$ demeure très exigeant.

Szymanowski et Ruszczynski (1972) considèrent l'approche de Polyak (1971) et tentent d'établir à chaque itération de l'algorithme des tests de précision afin de déterminer si une augmentation de la précision est nécessaire ou non. Puisque ces tests exigent possiblement la résolution de plusieurs problèmes de programmation quadratique et que la valeur optimale de $\underset{y \in \Omega}{\text{Max}}\ (C(y), x^k - y)$ n'est pas connue, ils ne sont pas applicables en pratique.

Une approche plus simple est donnée par Polyak (1967). Lorsque la fonction de coût C(x) est fortement monotone de paramètre $m>0$, et $\|C(x) - C(y)\| \leq L\|x - y\| \quad \forall\, x,y\in\Omega$ par exemple, la méthode suivante est prouvée convergente au taux d'une série géométrique:

$$x^{k+1}=P_{\Omega}(x^k - \alpha_k C(x^k)); \quad \alpha_k \text{ est tel que } 0<\lim_k \alpha_k < \frac{2m}{L^2},\ \alpha_k \geq 0,\ \sum_{k=1}^{\infty} \alpha_k = \infty.$$

Par contre, lorsque la condition de forte monotonie n'est pas satisfaite, une méthode de régularisation est utilisée:

$$x^{k+1} = P_{\Omega}(x^k - \alpha_k (C(x^k) + \varepsilon_k B(x^k)))$$

où C(x) est monotone, B(x) est uniformément monotone de paramètre $n>0$, C(x) et B(x) sont continus,

$$\|C(x) - C(y)\| \leq L\|x - y\|,\ \|b(x) - B(y)\| \leq L\|x - y\|, \quad \forall\, x,y\in\Omega\ ,$$

$\lim_{k\to\infty} [\alpha_k(1+\varepsilon_k)^2/\varepsilon_k] < 2n/L^2$ et $\{\alpha_k\},\{\varepsilon_k\}$ doivent satisfaire les conditions de la proposition 7.3.5.

Un choix possible de α_k, ε_k est: $\alpha_k = \theta\varepsilon_k$, $\varepsilon_k = k^{-p}$, $0<p<\frac{1}{2}$, $\theta < 2n/L^2$. On peut poser aussi $B(x) = x - a$, (où a est un vecteur constant) par exemple et considérer $L \leq 1$.

Des conditions autres que celles de Lipschitz sur C et B sont données par Bakushinskii et Polyak (1974), nous permettant des choix alternatifs au précédent.

De façon similaire à Bakushinskii et Polyak (1974), Auslender (1976) démontre la convergence de l'algorithme suivant vers un flot d'équilibre:

$$x^{k+1} = P_{\Omega}(x^k - \alpha_k C(x^k)), \qquad k = 0,1,2,\ldots$$

où $\alpha_k > 0$, $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \alpha^2 < \infty$, C(x) est bornée et fortement monotone.

Il fait remarquer de plus que l'hypothèse de forte monotonie n'est pas nécessaire lorsque la condition suivante est satisfaite: pour tout flot d'équilibre $x^* \in \Omega$, $g(x^*) = 0$ et $\arg \operatorname{Max}_{y \in \Omega} (C(y), x^* - y)$ est aussi un flot d'équilibre.

Deux cas:

1) La fonction de coût est strictement monotone

soit $x^* \in \Omega$, l'unique flot d'équilibre,

$$0 = \operatorname{Max}_{y \in \Omega} (C(y), x^* - y) = C(\bar{x}), x^* - \bar{x}) < (C(x^*), x^* - \bar{x}) \leq 0$$

si $\bar{x} \neq x^*$

$$\Rightarrow \bar{x} = x^*.$$

Par conséquent, la condition de forte monotonie peut être ramenée à celle de stricte monotonie.

2) La fonction de coût est monotone

soit x^* un flot d'équilibre,

si la condition "$\arg \operatorname{Max}_{y \in \Omega} (C(y), x^* - y)$ est un flot d'équilibre" est satisfaite, alors la convergence est aussi assurée.

Cette liste est loin d'être exhaustive mais elle donne un bon aperçu des différentes méthodes de projection applicables au problème de programmation convexe $\operatorname{Min}_{x \in \Omega} g(x)$.

c) Calcul d'un point de selle de $(C(x), x-y)$ sur $\Omega \times \Omega$

Nous allons maintenant considérer une approche de projection pour calculer un point de selle de la fonction $\phi(x,y) = (C(x), x-y)$ sur $\Omega \times \Omega$.

Observons le schéma itératif suivant:

$$x^{k+1} = P_{\Omega}(x_k - \alpha_k(C(x^k) + J_C(x^k)^t(x^k - y^k))$$

$$y^{k+1} = P_{\Omega}(y_k - \alpha_k C(x^k)).$$

Dans le cas où $(C(x), x-y)$ est convexe par rapport à x pour un y donné, et $\{x^* \in \Omega \mid \arg \underset{x \in \Omega}{\text{Min}} (C(x), x - y^*) = x^*, y^* \in \overline{\Omega}\} = \overline{\Omega}$, c'est-à-dire $\phi(x,y)$ est stable par rapport à x, la méthode a été examinée par Majstrovsky (1976). Ce papier est disponible en russe seulement.

L'hypothèse de stabilité de $\phi(x,y)$ par rapport à x est satisfaite lorsque $\phi(x,y)$ est strictement convexe par rapport à x pour un y donné.

Dem'yanov et Pevnyi (1972) ont montré la convergence de cette méthode lorsque $\alpha_k \equiv \alpha$ est choisi suffisamment petit, sous des conditions peu réalistes. Bakushinskii et Polyak (1974) modifient l'approche précédente par régularisation:

$$x^{k+1} = P_{\Omega}(x_k - \alpha_k(C(x^k) + J_C(x^k)^t(x^k - y^k) + \varepsilon_k \nabla f_1(x_k)))$$

$$y^{k+1} = P_{\Omega}(y_k - \alpha_k(C(x^k) + \varepsilon_k \nabla f_2(y_k)))$$

où $f_1(x)$, $f_2(y)$ sont fortement convexes,

α_k et ε_k satisfont les hypothèses de la proposition 7.3.5,

$(C(x), x-y)$ est convexe par rapport à x .

Aucune hypothèse de stricte convexité-stricte concavité ou de stabilité de $\phi(x,y)$ n'est nécessaire.

De façon plus explicite, Auslender (1976) donne une condition nécessaire et suffisante d'optimalité c'est-à-dire, lorsque (x^*,y^*) est un point de selle de ϕ sur $\Omega \times \Omega$. Il suppose que $(C(x), x-y)$ est convexe par rapport à $x \in \Omega$.

Proposition 7.3.6

Puisque $\phi(x,y^*)$ et $\phi(x^*,y)$ sont continues $\forall\ (x^*,y^*) \in \bar{\Omega} \times \bar{\Omega}$,

$$\bar{\Omega} \neq \phi,\ \phi(x^*,y^*) = 0 \quad \forall\ (x^*,y^*) \in \bar{\Omega} \times \bar{\Omega}$$

alors une condition nécessaire et suffisante pour que (x^*,y^*) soit un point de selle est qu'il existe un $c^* \in \partial_x \phi(x^*,y^*)$ et un $d^* \in \partial_y \phi(x^*,y^*)$ tels que:

$$P_\Omega(x^* - \rho c^*) = x^*$$

$$P_\Omega(x^* + \rho d^*) = y^* \ .$$

Puisque $\phi(x,y)$ est différentiable par rapport à x et à y, cette condition d'optimalité peut s'écrire:

$$P_\Omega(x^* - \rho(C(x^*) + J_C(x^*)^t(x^* - y^*))) = x^* \qquad (7.3.5)$$

$$P_\Omega(y^* - \rho C(x^*)) = y^* \qquad (7.3.6)$$

Or, dans le cas où les fonctions de coûts sont strictement monotones, $\bar{\Omega}$ est un singleton et la condition précédente s'écrit encore: $P_\Omega(x^* - \rho C(x^*)) = x^*$. Enfin, lorsque les fonctions de coût sont monotones, mais la fonction $(C(x), x-y)$ n'est pas nécessairement convexe en x pour un y donné, alors les

conditions d'optimalité (7.3.5) et (7.3.6) sont suffisantes et (7.3.6) est nécessaire.

A partir de la proposition 7.3.6, Auslender propose alors l'algorithme suivant:

"Soit $a>0$, $\{\rho_{1,k}\}$, $\{\rho_{2,k}\}$, $\{\lambda_k\}$ des suites de nombres positifs tels que

$\lim_{k\to\infty} \lambda_k = 0$,

$\sum_{k=1}^{\infty} \lambda_k = +\infty$, $(x_0,y_0) \in \Omega \times \Omega$, on construit une suite de points $\{(x_k,y_k)\}$ vérifiant les relations:

$$x_{k+1} = P_\Omega(a \cdot x_k - \rho_{1,k}(C(x^k) + J_C(x^k)^t(x^k - y^k))) \qquad (7.3.7)$$

$$y_{k+1} = P_\Omega(a \cdot y_k - \rho_{2,k}C(x^k)). \qquad (7.3.8)$$

L'algorithme s'arrête si $x_{k+1} = x_k$ et $y_{k+1} = y_k$."

A partir de la proposition 7.3.6, si $a=1$ et $x_{k+1} = x_k$, $y_{k+1} = y_k$ alors (x_k,y_k) est un point de selle de ϕ par rapport à $\Omega \times \Omega$. De même, Auslender nous fait remarquer que, si $a = 1 - 2\alpha\rho(\alpha>0)$, $\rho_{1,k} = \rho_{2,k} = \rho$ et si $x_{k+1} = x_k$ et $y_{k+1} = y_k$ alors (x_k,y_k) est un point de selle de la fonction $\phi_\alpha(x,y) = \phi(x,y) + \alpha(\|x\|^2 - \|y\|^2)$ sur $\Omega \times \Omega$. Malheureusement, ces deux approches sont montrées convergentes sous des conditions trop restrictives. Par exemple, dans le cas de fonctions de coûts affines, il considère le cas où le Jacobien est symétrique et défini-positif.

Plusieurs autres méthodes ont été laissées de côté parce qu'elles exigeaient elles aussi des hypothèses trop restrictives et/ou un procédé de

calcul difficilement implantable dans le cadre des problèmes d'équilibre du moins [McCormick et Tapia, (1970, 1972); Bertsekas, 1974; Dunn, 1981].

Remarquons en terminant le choix immense des méthodes de projection. Le but de la prochaine section est de généraliser l'approche de Dafermos (1980) dans le cadre du problème d'affectation.

7.4 Généralisation des méthodes de projection pour résoudre un problème d'équilibre général: Cas du flot sur les arcs

Supposons que les fonctions de coût sont strictement monotones, c'est-à-dire, il existe un flot d'équilibre unique. Nous pouvons alors écrire la condition nécessaire et suffisante d'optimalité qui suit:

Proposition 7.4.1

Soit F, G et H, trois matrices symétriques et définies positives, ρ_1, ρ_2 et ρ_3, trois nombres réels strictement positifs, alors

$$P_{\Omega}[x^* - \rho_1 F^{-1}(C(x^*) + \rho_2 G^{-1} J_C(x^*)^t(x^* - y^*))] = x^* \qquad (7.4.1)$$

$$P_{\Omega}[y^* - \rho_3 H^{-1} C(x^*)] = y^* \qquad (7.4.2)$$

si et seulement si $x^* = y^*$ est le flot d'équilibre.

Preuve

a) Supposons que $x^* = y^*$ est le flot d'équilibre unique, alors les conditions (7.4.1) et (7.4.2) peuvent s'écrire

$$P_{\Omega}[x^* - \rho_1 F^{-1} C(x^*)] = x^* \qquad (7.4.3)$$

$$P_{\Omega}[x^* - \rho_3 H^{-1} C(x^*)] = x^* \qquad (7.4.4)$$

Enfin, on a (7.4.3) $\Leftrightarrow$ $(C(x^*), x - x^*) \geq 0 \quad \forall\, x \in \Omega$

et (7.4.4) $\Leftrightarrow$ $(C(x^*), x - x^*) \geq 0 \quad \forall\, x \in \Omega$.

b) Supposons que les relations (7.4.1) et (7.4.2) sont satisfaites, nous obtenons après transformation (voir proposition 7.2.1):

$$(C(x^*) + \rho_2 G^{-1} J_C(x^*)^t (x^* - y^*), x - x^*) \geq 0 \quad \forall\, x \in \Omega \qquad (7.4.5)$$

$$(C(x^*), y - y^*) \geq 0 \quad \forall\, y \in \Omega \; . \qquad (7.4.6)$$

En posant $x = y^*$ et $y = x^*$ dans les relations (7.4.5) et (7.4.6) respectivement, et en additionnant ces deux inéquations, on peut écrire:

$$\rho_2 G^{-1} (J_C(x^*)^t (x^* - y^*), y^* - x^*) \geq 0$$

c'est-à-dire, $(J_C(x^*)^t (x^* - y^*), y^* - x^*) \geq 0$.

Or, Auslender (1976) démontre l'équivalence suivante:

$C(x)$ est strictement monotone sur Ω

si et seulement si $(J_C(x)^t (x - y), x - y) > 0 \quad \forall\, x \neq y \in \Omega$

D'où, $x^* = y^*$ et les relations (7.4.1) et (7.4.2) deviennent les relations (7.4.3) et (7.4.4). Par conséquent, $x^* = y^*$ est le flot d'équilibre. □

La proposition 7.4.1 suggère l'algorithme suivant:

<u>Etape 0</u>. Soit $\rho_1, \rho_2, \rho_3 > 0$; F, G, H symétriques et définies positives, soit $x^0, y^0 \in \Omega$ données.

Posons $k \leftarrow 0$.

Etape 1. Calculer

$$x^{k+1} = P_{\Omega}[x^k - \rho_1 F^{-1}(C(x^k) + \rho_2 G^{-1} J_C(x^k)^t (x^k - y^k))] \qquad (7.4.7)$$

$$y^{k+1} = P_{\Omega}[y^k - \rho_3 H^{-1} C(x^k)] \qquad (7.4.8)$$

Poser $k \leftarrow k+1$ et aller à l'étape 1.

L'algorithme s'arrête lorsque $x^{k+1} = x^k$ et $y^{k+1} = y^k$.

Il reste à montrer que cet algorithme génère un point fixe. Pour ce faire, nous allons nous baser essentiellement sur l'approche utilisée par Dafermos (1980). Nous supposons la forte monotonie des fonctions de coût.

Proposition 7.4.2

Soit $\bar{\gamma}$ le maximum sur Ω de la valeur propre maximale de la matrice symétrique définie positive

$$[J_C(x)]\ F^{-1}[J_C(x)]^t \qquad (7.4.9)$$

en posant $\rho_1 = \rho_3$, $F = H$ et $G = I$ la matrice identité, en choisissant ρ_1 et ρ_2 telle que $\rho_1\rho_2 < 2\alpha/\bar{\gamma}$ où α est le paramètre de forte monotonie, nous avons:

$$\lim_{k\to\infty} (C(x^k),\ x^k - y^k) = 0$$

et de plus, $\|x^{k+1} - y^{k+1}\|_F \leq \lambda \|x^k - y^k\|_F$ où $\lambda = \{1-(1/\mu)\rho_1\rho_2(2\alpha - \bar{\gamma}\rho_1\rho_2)\}^{\frac{1}{2}}$

$\forall\ k \geq 0$, μ étant la valeur propre maximale de F, $\|x\| = (x, Fx)^{\frac{1}{2}}$.

Preuve:

Soient $\{x^k\}_{k=0}^{\infty}$, $\{y^k\}_{k=0}^{\infty}$ deux suites de points générés par l'algorithme précédent.

A l'aide de la proposition 7.2.1, on peut écrire:

$$([(x^{k+1}-x^k)+\rho_1 F^{-1}(C(x^k)+\rho_2 J_C(x^k)^t(x^k-y^k))],\ x-x^{k+1}) \geq 0 \quad \forall\, x\in\Omega \qquad (7.4.10)$$

$$([(y^{k+1}-y^k)+\rho_1 F^{-1}C(x^k))],\ y-y^{k+1}) \geq 0 \quad \forall\, y\in\Omega \qquad (7.4.11)$$

En posant $x=y^{k+1}$ et $y=x^{k+1}$ respectivement dans les relations (7.4.10) et (7.4.11) et en additionnant les deux inéquations, nous obtenons

$$(x^{k+1}-y^{k+1},\ F(x^{k+1})-y^{k+1})) \leq ([F(y^k-x^k)+\rho_1\rho_2 J_C(x^k)^t(x^k-y^k))],\ y^{k+1}-x^{k+1})$$

c'est-à-dire

$$\|x^{k+1}-y^{k+1}\|_F^2 \leq (\{(y^k-x^k)+\rho_1\rho_2 F^{-1}\ J_C(x^k)^t(x^k-y^k)\},\ F(y^{k+1}-x^{k+1})).$$

En appliquant l'inégalité de Schwartz, cela devient

$$\|x^{k+1}-y^{k+1}\|_F \leq \|(y^k-x^k)+\rho_1\rho_2 F^{-1}\ J_C(x^k)^t(x^k-y^k)\|_F\ ;$$

puis, en élevant au carré et en développant le côté droit, il arrive que

$$\|x^{k+1}-y^{k+1}\|_F^2 \leq \|x^k-y^k\|_F^2+\rho_1^2\rho_2^2(x^k-y^k)^t J_C(x^k)\ F^{-1}\ J_C(x^k)^t(x^k-y^k)$$

$$-2\rho_1\rho_2(x^k-y^k,\ J_C(x^k)^t(x^k-y^k)). \qquad (7.4.12)$$

Sachant que C(x) est fortement monotone de rayon $\alpha>0$, F est symétrique et définie positive,

$$\|x^{k+1} - y^{k+1}\|_F^2 \leq \|x^k - y^k\|_F^2 + \rho_1^2\rho_2^2\bar{\gamma}(x^k - y^k,\ x^k - y^k)$$

$$- 2\rho_1\rho_2\,\alpha(x^k - y^k,\ x^k - y^k)$$

$$= \|x^k - y^k\|_F^2 - \rho_1\rho_2[2\alpha - \bar{\gamma}\rho_1\rho_2]\,(x^k - y^k,\ x^k - y^k)\ .$$

Nous choisissons ρ_1 et ρ_2 telles que $\rho_1\rho_2 < 2\alpha/\bar{\gamma}$. Etant donné que $\|x^k - y^k\|_F^2 \leq \mu(x^k - y^k,\ x^k - y^k)$, où μ est la valeur propre maximale de F, nous obtenons enfin

$$\|x^{k+1} - y^{k+1}\|_F^2 \leq \{1 - (\tfrac{1}{\mu})\rho_1\rho_2(2\alpha - \bar{\gamma}\rho_1\rho_2)\}\ ,\ \|x^k - y^k\|_F^2 \qquad (7.4.13)$$

La relation (7.4.13) indique la présence d'une contraction relativement à la norme $\|x\| = (x, Fx)^{\frac{1}{2}}$. D'où

$$\lim_{k\to\infty} (C(x^k),\ x^k - y^k) = 0 \qquad (7.4.14)$$

Enfin, en posant $\lambda = \{1 - (1/\mu)\rho_1\rho_2(2\alpha - \bar{\gamma}\rho_1\rho_2)\}^{\frac{1}{2}}$, la relation suivante est obtenue:

$$\|x^k - y^k\|_F \leq \lambda^k M \quad \text{où} \quad M = \|x^0 - y^0\|_F \quad \text{est fixé.}$$

□

Si l'on désire la meilleure vitesse de convergence, on devra choisir $\rho_1\rho_2 = \alpha/\bar{\gamma}$, de sorte que $\lambda = (1 - \alpha^2/\mu\bar{\gamma})^{\frac{1}{2}}$. Par exemple, on peut prendre $\rho_1 = \alpha/\bar{\gamma}$ et $\rho_2 = 1$.

Nous allons maintenant montrer que l'algorithme précédent génère véritablement un point fixe.

Proposition 7.4.3

Supposons que les conditions de la proposition 7.4.2 sont satisfaites et que la continuité du Jacobien de la fonction de coût est assurée, alors l'algorithme précédent génère un point fixe à la condition que $\rho_1 < 2\alpha/\gamma$, où γ est le maximum sur Ω de la valeur propre maximale de la matrice symétrique définie positive

$$[J_C(x)]^t F^{-1} [J_C(x)]. \qquad (7.4.15)$$

Preuve

Soient $\{x^k\}_{k=0}^{\infty}$, $\{y_k\}_{k=0}^{\infty}$ et $\{\bar{x}^k\}_{k=0}^{\infty}$, $\{\bar{y}^k\}_{k=0}^{\infty}$ des suites de points possiblement générées par cet algorithme.

A l'aide de la proposition 7.2.1, on peut écrire:

$$([x^{k+1} - x^k + \rho_1 F^{-1}(C(x^k) + \rho_2 J_C(x^k)^t (x^k - y^k))], \; x - x^{k+1}) \geq 0$$

$$\forall \, x \in \Omega \qquad (7.4.16)$$

$$([\bar{x}^{k+1} - \bar{x}^k + \rho_1 F^{-1}(C(\bar{x}^k) + \rho_2 J_C(\bar{x}^k)^t (\bar{x}^k - \bar{y}^k))], \; x - \bar{x}^{k+1}) \geq 0$$

$$\forall \, x \in \Omega \qquad (7.4.17)$$

En posant $x = \bar{x}^{k+1}$ et $x = x^{k+1}$ dans (7.4.16) et (7.4.17) respectivement, et en additionnant les deux inéquations, nous obtenons:

$$\|x^{k+1} - \bar{x}^{k+1}\|_F^2 \leq [(\bar{x}^k - x^k) + \rho_1 F^{-1}(C(x^k) - C(\bar{x}^k) + \rho_2 (J_C(x^k)^t (x^k - y^k)$$

$$- J_C(\bar{x}^k)^t (\bar{x}^k - \bar{y}^k)))]^t \cdot F(\bar{x}^{k+1} - x^{k+1}) \; .$$

En appliquant l'inégalité de Schwartz, cela devient

$$\|x^{k+1}-\bar{x}^{k+1}\|_F \le \|\bar{x}^k-x^k+\rho_1 F^{-1}(C(x^k)-C(\bar{x}^k)+\rho_2(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k-\bar{y}^k)))\|_F \ ;$$

en élevant au carré et en développant le côté droit de cette inégalité, il arrive que

$$\begin{aligned}\|x^{k+1}-\bar{x}^{k+1}\|_F^2 \le \|x^k-\bar{x}^k\|_F^2 &+\rho_1^2(C(x^k)-C(\bar{x}^k))^t F^{-1}(C(x^k)-C(\bar{x}^k))-2\rho_1(C(x^k)-C(\bar{x}^k),\ x^k-\bar{x}^k)\\ &+\rho_1^2\rho_2^2(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))^t F^{-1}(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\\ &-2\rho_1\rho_2(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))^t(x^k-\bar{x}^k)\\ &+2\rho_1^2\rho_2(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))^t F^{-1}(C(x^k)-C(\bar{x}^k)).\end{aligned}$$

En se servant encore de l'inégalité de Schwartz, on obtient:

$$\begin{aligned}\|x^{k+1}-\bar{x}^{k+1}\|_F^2 \le \|x^k-\bar{x}^k\|_F^2 &+\rho_1^2(C(x^k)-C(\bar{x}^k))^t F^{-1}(C(x^k)-C(\bar{x}^k))-2\rho_1(C(x^k)-C(\bar{x}^k),x^k-\bar{x}^k) \quad (7.4.18)\\ &+\rho_1^2\rho_2^2\|F^{-1}(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F^2\\ &+2\rho_1\rho_2\|F^{-1}(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F\cdot\|x^k-\bar{x}^k\|_F\\ &+2\rho_1^2\rho_2\|F^{-1}(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F\cdot F^{-1}(C(x^k)-C(\bar{x}^k))\|_F .\end{aligned}$$

Déterminons maintenant une borne supérieure à la quantité

$$\|F^{-1}(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F . \qquad (7.4.19)$$

Puisque le Jacobien des fonctions de coût est continu et par conséquent borné, on peut poser $\tilde{\gamma} = \underset{x\in\Omega}{\text{Max}} \|J_C(x)^t\|_F$ où $\|J_C(x)^t\|_F$ est la plus petite valeur de N(x) satisfaisant la relation $\|J_C(x)^t y\| \leq N(x)\|y\|_F \quad \forall\, y \in \Omega$.

D'où

$$\|F^{-1}(J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F \leq (\frac{1}{\bar{\mu}})\|J_C(x^k)^t(x^k-y^k)-J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k)\|_F$$

où $\bar{\mu}$ est la valeur propre minimale de F, ce qui donne encore

$$\leq (\frac{1}{\bar{\mu}})\{\|J_C(x^k)^t(x^k-y^k)\|_F+\|J_C(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k)\|_F\}; \tag{7.4.19}$$

et d'après l'inégalité du triangle, on obtient de nouveau

$$\leq (\frac{\tilde{\gamma}}{\bar{\mu}})\{\|x^k-y^k\|_F+\|\bar{x}^k-\bar{y}^k\|_F\}, \tag{7.4.19}$$

$$\leq (\frac{\tilde{\gamma}}{\bar{\mu}})\lambda^k\{\|x^0-y^0\|_F+\|\bar{x}^0-\bar{y}^0\|_F\}\ ; \text{ puis,} \tag{7.4.19}$$

d'après la proposition 7.4.2, nous avons

$$\leq a\,\lambda^k \quad \text{où} \quad a = (\frac{\tilde{\gamma}}{\bar{\mu}})\{\|x^0-y^0\|_F+\|\bar{x}^0-\bar{y}^0\|_F\} \tag{7.4.19}$$

est une constante. (7.4.20)

Remarquons que cette quantité décroît strictement vers 0 à mesure que k croît.

De plus,

$\|F^{-1}(C(x^k)-C(\bar{x}^k))\|_F^2 = (C(x^k)-C(\bar{x}^k),\ F^{-1}(C(x^k)-C(\bar{x}^k)))$ par définition de $\|\cdot\|_F$,

$\leq (x^k-\bar{x}^k,\ [\partial c/\partial x]^t F^{-1}[\partial c/\partial x](x^k-\bar{x}^k))$ d'après l'appendice 2 de Dafermos (1980).

$$\leq \gamma(x^k - \bar{x}^k \ , \ x^k - \bar{x}^k) \qquad \text{d'après (7.4.9)}$$

$$\leq (\gamma/\bar{u})\|x^k - \bar{x}^k\|_F^2 \qquad \text{où } \bar{u} \text{ est la valeur propre minimale de F.} \qquad (7.4.21)$$

A partir de (7.4.20), (7.4.21) et du fait que la fonction de coût est fortement monotone de rayon $\alpha > 0$, l'inégalité (7.4.18) peut maintenant s'écrire plus simplement:

$$\|x^{k+1} - \bar{x}^{k+1}\|_F^2 \leq \{1-(1/u)\rho_1(2\alpha-\gamma\rho_1)\}\cdot\|x^k - \bar{x}^k\|_F^2$$
$$+a\lambda^k\cdot\{\rho_1^2\rho_2^2 a\lambda^k + 2\rho_1\rho_2\|x^k - \bar{x}^k\|_F \qquad (7.4.22)$$
$$+2\rho_1^2\rho_2(\gamma/\bar{u})^{\frac{1}{2}}\|x^k - \bar{x}^k\|_F$$

Posons $\bar{\lambda} = \{1-(1/u)\rho_1(2\alpha-\gamma\rho_1)\}$. Sachant que Ω est un ensemble compact, alors la quantité $\|x^k - \bar{x}^k\|_F$ est bornée pour tout $k \geq 0$ de sorte qu'il existe une borne supérieure b finie aux éléments de la suite

$$\{a\cdot(\rho_1^2\rho_2^2 a\lambda^k + 2\rho_1\rho_2\|x^k - \bar{x}^k\|_F + 2\rho_1^2\rho_2(\gamma/\bar{u})^{\frac{1}{2}}\|x^k - \bar{x}^k\|_F)\}_{k=0}^{\infty} .$$

Par conséquent (7.4.22) peut encore s'écrire:

$$\|x^{k+1} - \bar{x}^{k+1}\|_F^2 \leq \bar{\lambda}\cdot\|x^k - \bar{x}^k\|_F^2 + b\lambda^k. \qquad (7.4.23)$$

En procédant par induction, on obtient de (7.4.23):

$$\|x^{k+1} - \bar{x}^{k+1}\|_F^2 \leq \bar{\lambda}^{k+1}\cdot\|x^0 - \bar{x}^0\|_F^2 + b\ k\tilde{\lambda}^k \text{ où } \tilde{\lambda}=\text{Max}\{\lambda,\bar{\lambda}\}. \qquad (7.4.24)$$

Choisissons $\rho_1 < 2\alpha/\gamma$. Ainsi, $\bar{\lambda}, \tilde{\lambda} \in (0,1)$.

Lorsque k croît, le côté droit de l'inégalité (7.4.24) tend vers 0.

Par conséquent, $\|x^k - \bar{x}^k\|_F \xrightarrow[k\to\infty]{} 0$.

Ce résultat nous permet de conclure que l'algorithme génère un point fixe pour la relation (7.4.7). De plus, en regard de la proposition (7.4.2), le résultat de la proposition (7.4.3) s'ensuit. □

Réécrivons maintenant le schéma itératif précédent dans le cadre des propositions (7.4.2) et (7.4.3):

$$x^{k+1} = P_\Omega(x^k - \rho_1 F^{-1}(C(x^k) + \rho_2 J_C(x^k)^t(x^k - y^k))), \tag{7.4.25}$$

$$y^{k+1} = P_\Omega(y^k - \rho_1 F^{-1} C(x^k)); \quad k=0,1,2,\ldots \tag{7.4.26}$$

On remarque que les deux conditions $\rho_1\rho_2 < 2\alpha/\bar{\gamma}$ et $\rho_1 < 2\alpha/\gamma$ sont satisfaites lorsque $\rho_1 < \frac{2\alpha}{\gamma}$ et $\rho_2 \leq \gamma/\bar{\gamma}$. De plus, afin d'accélérer la convergence de l'algorithme, un choix judicieux de ces paramètres est à envisager comme par exemple, $\rho_1 = \frac{\alpha}{\gamma}$ et $\rho_2 = \gamma/\bar{\gamma}$.

Enfin, en prenant $x^0 = y^0$ et/ou $\rho_2 = 0$, on obtient l'algorithme de Dafermos (1980).

Un cas particulier très intéressant est celui où les fonctions de coût sont affines; (7.4.25) et (7.4.26) s'écrivent alors:

$$x^{k+1} = P_\Omega(x^k - \rho_1 F^{-1}(C(x^k) + \rho_2 A^t(x^k - y^k)) \tag{7.4.27}$$

$$y^{k+1} = P_\Omega(y^k - \rho_1 F^{-1} C(x^k)) \tag{7.4.28}$$

où $C(x) = Ax + b$.

Considérons maintenant la résolution des problèmes (7.4.10) et (7.4.11). Il s'agit de déterminer un point d'équilibre lorsque les fonctions de coût sont affines et le Jacobien correspondant est symétrique et défini positif. Ces fonctions de coût sont de la forme $Fx+(\rho_1 C(x^k)+\rho_1\rho_2 J_C(x^k)^t(x^k-y^k)-Fx^k)$ et $Fy+(\rho_1 C(x^k)-Fy^k)$ respectivement.

Un dernier point reste à traiter. Nous voulons un estimé du taux de convergence.

Proposition 7.4

Soit x* l'unique flot d'équilibre. Si (x^0,y^0) est le point initial de l'algorithme, alors

$$\|x^k-x^*\|_F^2 \leq \bar{\lambda}^k \cdot \|x^0-x^*\|_F^2 + bk\tilde{\lambda}^k . \qquad (7.4.29)$$

De plus, $$\|y^k-x^*\|_F \leq \|x^k-x^*\|_F + \lambda^{k/2}\|x^0-y^0\|_F . \qquad (7.4.30)$$

Preuve

En posant dans la proposition 7.4.3, $\bar{x}^0 = x^* = \bar{y}^0$, l'algorithme génère la suite de points $\{(\bar{x}^k,\bar{y}^k)\}_{k=0}^{\infty}$ tels que $(\bar{x}^k,\bar{y}^k) = (x^*,x^*)\ \forall\ k\geq 0$, cela en supposant que le critère d'arrêt de l'algorithme est omis et qu'on laisse le processus se dérouler indéfiniment. Il s'agit alors de poser $\bar{x}^0 = x^* = \bar{y}^0$ dans la relation (7.4.24) pour obtenir (7.4.29).

La relation (7.4.30) découle directement de (7.4.29), de la proposition 7.4.2 et de l'inégalité de Schwartz.

□

Dans la prochaine section, nous examinerons le comportement des principaux algorithmes de projection rencontrés depuis le début de ce chapitre à l'aide d'un petit exemple.

7.5 Essais effectués à partir d'un exemple de Dafermos (1980a).

Examinons brièvement le comportement des algorithmes précédents sur un tout petit réseau [Dafermos, 1980].

Considérons le réseau simple suivant composé des arcs a_1, a_2, a_3, b_1 et b_2.

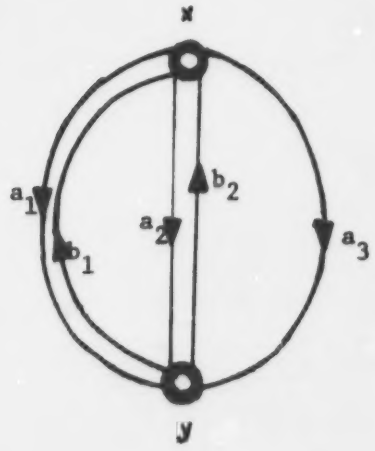


Figure 7.5.1 - Réseau de Dafermos (1980)

Le vecteur de demande est: $d_{xy} = 210$ et $d_{yx} = 120$.

Les fonctions de coût sont: $c_{a_1} = 10x_{a_1} + 5x_{b_1} + 1000$

$c_{a_2} = 15x_{a_2} + 5x_{b_2} + 950$

$c_{a_3} = 20x_{a_3} + 3000$

$c_{b_1} = 20x_{b_1} + 2x_{a_1} + 1000$

$c_{b_2} = 25x_{b_2} + x_{a_2} + 1300.$

La solution d'équilibre est $x^* = (120, 90, 0, 70, 50)$.

Un calcul simple nous donne comme paramètre de forte monotonie $\alpha = 8.9$. Choisissons $F = \text{diag}[10,15,20,20,25]$. Nous obtenons alors $\gamma = 29.8$ relativement à la matrice (7.4.15).

Considérons premièrement les fonctions de coût associées à (7.4.27) sous la forme

$$Fx+h^k \quad \text{où} \quad h^k = [\rho_1(1+\rho_2)C(x^k)-\rho_1\rho_2 C(y^k)-Fx^k] ,$$

puis, celles associées à (7.4.28) sous la forme

$$Fy+g^k \quad \text{où} \quad g^k = [\rho_1 C(x^k)-Fy^k] .$$

Les deux problèmes d'équilibre correspondants ont été résolus explicitement par Dafermos (1980). La solution est la suivante:

$$z_{a_1} = 96.9 - 0.0538f_{a_1} + 0.0308f_{a_2} + 0.0230f_{a_3}$$

$$z_{a_2} = 64.6 + 0.0308f_{a_1} - 0.0462f_{a_2} + 0.0154f_{a_3}$$

$$z_{a_3} = 48.5 + 0.0230f_{a_1} + 0.0154f_{a_2} - 0.0384f_{a_3}$$

$$z_{b_1} = 66.7 + 0.0222(f_{b_2} - f_{b_1})$$

$$z_{b_2} = 53.3 - 0.0222(f_{b_2} - f_{b_1})$$

lorsque $z_{a_3} > 0$, ou par

$$z_{a_1} = 126 + 0.04(f_{a_2} - f_{a_1})$$

$$z_{a_2} = 84 - 0.04(f_{a_2} - f_{a_1})$$

$$z_{a_3} = 0$$

$$z_{b_1} = 66.7 + 0.0222(f_{b_2} - f_{b_1})$$

$$z_{b_2} = 53.3 - 0.0222(f_{b_2} - f_{b_1}), \text{ dans le cas contraire.}$$

Lorsque le problème (7.4.27) est à résoudre, les vecteurs z et f sont remplacés par x^{k+1} et h^k respectivement, tandis que le problème (7.4.28) fait intervenir les vecteurs y^{k+1} et g^k respectivement.

Posons $x^0 = (70,70,70,60,60)$, $y^0 = (80,70,60,60,60)$. Le résultat des calculs est donné à la Table 7.5.1.

Certes, cet exemple seul est beaucoup trop petit pour nous permettre de conclure quoi que ce soit à propos de l'algorithme de la section 4. Il est difficile de voir l'influence des solutions initiales x^0 et y^0 et de la valeur des paramètres ρ_1 et ρ_2. Par contre, la suite $\{y^k\}$ semble converger légèrement plus lentement que la suite $\{x^k\}$. Cela correspond bien au résultat de la proposition 7.4.4.

L'algorithme de Dafermos et notre algorithme général se comparent sensiblement au point de vue de la vitesse de convergence (voir Tableau 7.5.1). Malheureusement, ce dernier exige la résolution de deux sous-problèmes au lieu d'un seul. Par conséquent, il semble préférable d'utiliser celui de Dafermos, un cas spécial de notre algorithme général. Deux

variantes de l'algorithme de la section 4 ont été démontrées convergentes expérimentalement. Elles consistent à remplacer $C(x^k)$ par $C(x^{k+1})$ et y^k par x^{k+1} respectivement dans la relation (7.4.8). Des résultats au moins aussi valables sont obtenus (tableau 7.5.2).

Dans le tableau 7.5.3, nous avons considéré le schéma itératif: $x^{k+1}=P_\Omega(x^k-\alpha_k F^{-1}C(x^k))$, $k=0,1,2,\ldots$ où $\alpha_k = 1/k+1$ et $\alpha_k = (C(x^k) - C(x^{k-1}), x^k-x^{k-1})/\|F^{-1}(C(x^k)-C(x^{k-1}))\|_F^2$ tandis que dans le tableau 7.5.4, ce sont les algorithmes énoncés aux propositions 7.3.1 et 7.3.2.

Nous remarquons facilement que les algorithmes des tableaux 7.5.1 à 7.5.3 et ceux du tableau 7.5.4 forment deux classes bien distinctes. Ceux du tableau 7.5.4 semblent vraiment impraticables; leur vitesse de convergence est beaucoup trop lente. Par contre, c'est l'algorithme énoncé à la proposition 7.3.2 lorsque $\lambda_k \equiv 1.999999 \simeq 2.0 \quad \forall\, k \geq 0$ qui est de loin le plus performant dans ceux de la table 7.5.4.

Enfin, l'algorithme qui semble le plus performant est celui de Sibony à la section 2 (voir tableau 7.5.3). Il donne une solution optimale à une décimale près après seulement quatre itérations. Un autre avantage sérieux est que α_k se calcule à l'itération k à partir des points x^k et x^{k-1} seulement. Il n'est pas nécessaire, comme dans l'algorithme de Dafermos (1980) par exemple, de satisfaire la condition $\alpha_k < 2\alpha/\gamma$ pour tout k; cette constante n'est pas connue généralement.

Tableau 7.5.1 - Algorithme de projection de la section 7.4

Type d'algorithme	ρ_1	ρ_2	Iter. No k	Type de flots	Arc A1	Arc A2	Arc A3	Arc B1	Arc B2
Dafermos $x^0 = y^0$	0.3	0.0	0	x	70.00	70.00	70.00	60.00	60.00
			1	x	89.35	76.92	43.72	63.56	56.43
			2	x	102.40	82.18	25.41	65.85	54.14
			3	x	111.23	86.13	12.63	67.31	52.68
			4	x	117.20	89.06	3.72	68.24	51.75
			5	x	119.82	90.17	0.00	68.83	51.16
			6	x	120.01	89.98	0.00	69.21	50.78
			7	x	120.10	89.89	0.00	69.48	50.51
			8	x	120.13	89.86	0.00	69.66	50.33
			9	x	120.13	89.86	0.00	69.79	50.20
			10	x	120.11	89.88	0.00	69.88	50.11
Général	0.3	1.0		x	70.00	70.00	70.00	60.00	60.00
			0	y	80.00	70.00	60.00	60.00	60.00
				x	92.34	76.92	40.73	63.90	56.09
			1	y	99.33	76.92	33.74	63.56	56.43
				x	106.53	82.23	21.23	66.18	53.81
			2	y	111.42	82.21	16.35	65.71	54.28
				x	115.52	86.21	8.26	67.51	52.48
			3	y	118.95	86.19	4.84	67.02	52.97
				x	121.03	88.96	0.00	68.29	51.70
			4	y	122.02	87.97	0.00	67.83	52.16
				x	121.20	88.79	0.00	68.74	51.25
			5	y	121.91	88.08	0.00	68.35	51.64
				x	121.19	88.80	0.00	69.05	50.94
			6	y	121.70	88.29	0.00	68.74	51.25
				x	121.09	88.90	0.00	69.28	50.71
			7	y	121.46	88.53	0.00	69.03	50.96
				x	120.95	89.04	0.00	69.45	50.54
			8	y	121.21	88.78	0.00	69.25	50.74
				x	120.80	89.19	0.00	69.58	50.41
			9	y	120.99	89.00	0.00	69.42	50.57
				x	120.66	89.33	0.00	69.69	50.30
			10	y	120.80	89.19	0.00	69.56	50.43

Tableau 7.5.2 - Variante de l'algorithme de projection de la section 7.4

Type d'algorithme	ρ_1	ρ_2	Iter. No k	Type de flots	Arc A1	Arc A2	Arc A3	Arc B1	Arc B2
Variante 1	0.3	1.0	0	x	70.00	70.00	70.00	60.00	60.00
				y	80.00	70.00	60.00	60.00	60.00
			1	x	92.34	76.92	40.73	63.90	56.09
				y	92.15	75.29	42.55	62.14	57.85
			2	x	104.32	81.78	23.88	65.56	54.43
				y	100.47	79.32	30.20	63.66	56.33
			3	x	111.37	85.14	13.47	66.46	53.53
				y	106.54	82.45	21.00	64.84	55.15
			4	x	115.90	87.52	6.57	67.08	52.91
				y	111.16	84.93	13.89	65.79	54.20
			5	x	119.02	89.28	1.69	67.57	52.42
				y	114.77	86.95	8.26	66.56	53.43
			6	x	120.19	89.80	0.00	67.97	52.02
				y	117.97	88.86	3.15	67.19	52.80
			7	x	120.25	89.74	0.00	68.33	51.66
				y	119.99	90.00	0.00	67.72	52.27
			8	x	120.27	89.72	0.00	68.66	51.33
				y	120.07	89.92	0.00	68.15	51.84
			9	x	120.27	89.72	0.00	68.92	51.07
				y	120.11	89.88	0.00	68.50	51.49
			10	x	120.25	89.74	0.00	69.13	50.86
				y	120.14	89.85	0.00	68.78	51.21

$x^{k+1}=P_\Omega(x^k-\rho_1 F^{-1}(C(x^k)+\rho_2 J_C(x^k)^t(x^k-y^k)))$

$y^{k+1}=P_\Omega(y^k-\rho_1 F^{-1}C(x^{k+1}))$

Tableau 7.5.2 (suite)

Type d'algorithme	ρ_1	ρ_2	Iter. No k	Type de flots	Arc A1	Arc A2	Arc A3	Arc B1	Arc B2
Variante 2	0.3	0.1	C	x	70.00	70.00	70.00	60.00	60.00
				y	80.00	70.00	60.00	60.00	60.00
			1	x	92.34	76.92	40.73	63.90	56.09
				y	111.63	83.84	14.51	67.46	52.53
			2	x	110.37	84.18	15.43	67.52	52.47
				y	121.33	88.66	0.00	69.66	50.33
			3	x	119.95	89.00	1.04	69.24	50.75
				y	120.98	89.01	0.00	70.10	49.89
			4	x	120.64	89.35	0.00	69.79	50.20
				y	120.56	89.43	0.00	70.03	49.96
			5	x	120.46	89.53	0.00	69.93	50.06
				y	120.29	89.70	0.00	70.01	49.98
			6	x	120.28	89.71	0.00	69.99	50.00
				y	120.15	89.84	0.00	70.03	49.96
			7	x	120.16	89.83	0.00	70.02	49.97
				y	120.07	89.92	0.00	70.04	49.95
			8	x	120.08	89.91	0.00	70.04	49.95
				y	120.03	89.96	0.00	70.06	49.93
			9	x	120.04	89.95	0.00	70.06	49.93
				y	120.00	89.99	0.00	70.07	49.92
			10	x	120.01	89.98	0.00	70.07	49.92
				y	119.99	90.00	0.00	70.08	49.91

$$x^{k+1} = P_\Omega(x^k - \rho_1 F^{-1}(C(x^k) + \rho_2 J_C(x^k)^t (x^k - y^k)))$$
$$y^{k+1} = P_\Omega(x^{k+1} - \rho_1 F^{-1} C(x^k))$$

Tableau 7.5.3 - Algorithme proposé par Sibony à la section 7.2

Type d'algorithme	ρ_1	ρ_2	Iter. No k	Type de flots	Arc A1	Arc A2	Arc A3	Arc B1	Arc B2
$x^0 = y^0$	1	0.0	0	x	70.00	70.00	70.00	60.00	60.00
	1/2		1	x	124.00	86.00	0.00	71.80	48.19
	1/3		2	x	121.63	88.36	0.00	70.79	49.20
	1/4		3	x	120.98	89.01	0.00	70.52	49.47
	1/5		4	x	120.68	89.31	0.00	70.40	49.59
	1/6		5	x	120.51	89.48	0.00	70.34	49.65
	1/7		6	x	120.40	89.59	0.00	70.31	49.68
	1/8		7	x	120.33	89.66	0.00	70.29	49.70
	1/9		8	x	120.27	89.72	0.00	70.28	49.71
	1/10		9	x	120.23	89.76	0.00	70.28	49.71
	1/11		10	x	120.19	89.80	0.00	70.28	49.71
$x^0 = y^0$	0.3	0.0	0	x	70.00	70.00	70.00	60.00	60.00
	0.976312		1	x	89.35	76.92	43.72	63.56	56.43
	0.970138		2	x	122.40	87.59	0.00	70.92	49.07
	0.863220		3	x	119.71	90.28	0.00	69.90	50.09
	0.849988		4	x	119.99	90.00	0.00	70.03	49.96
(1)	0.161530		5	x	119.98	90.01	0.00	70.03	49.96
	0.871766		6	x	119.98	90.01	0.00	70.03	49.96
	0.871766		7	x	119.98	90.01	0.00	70.03	49.96
	0.871766		8	x	119.98	90.01	0.00	70.03	49.96
	0.871766		9	x	119.98	90.01	0.00	70.03	49.96
	0.871766		10	x	119.98	90.01	0.00	70.03	49.96

(1) $$\rho_1^k = \frac{(C(x^k)-C(x^{k-1}) \ , \ x^k-x^{k-1})}{(C(x^k)-C(x^{k-1}) \ , \ G^{-1}(C(x^k)-C(x^{k-1})))}$$

Tableau 7.5.4 - Méthode de sous-gradient de Polyak

Type d'algorithme	Iter. No k	λ_k	α_k	Arc A1	Arc A2	Arc A3	Arc B1	Arc B2
$x^{k+1}=P_\Omega(x^k-\alpha_k(C(x^k)+J_C(x^k)^t(x^k-u^k)))$	0	1	0.000121	70.00	70.00	70.00	60.00	60.00
	1	1/2	0.000061	70.33	69.97	69.68	60.25	59.74
où	2	1/3	0.000041	70.50	69.96	69.53	60.38	59.61
u^k est le p.c.c. associé à $C(x^k)$	3	1/4	0.000030	70.61	69.95	69.42	60.47	59.52
	4	1/5	0.000024	70.69	69.95	69.35	60.53	59.46
$\alpha_k=\lambda_k/\|C(x^k)+J_C(x^k)^t(x^k-u^k)\|$	5	1/6	0.000020	70.76	69.95	69.28	60.59	59.40
	6	1/7	0.000017	70.81	69.94	69.23	60.63	59.36
$\lambda_k \to 0$	7	1/8	0.000015	70.86	69.94	69.19	60.67	59.32
	8	1/9	0.000014	70.90	69.94	69.15	60.70	59.29
	9	1/10	0.000012	70.94	69.93	69.12	60.73	59.26
$\sum_{k=1}^{\infty} \lambda_k = \infty$	10	1/11	0.000011	70.97	69.93	69.08	60.75	59.24
	100	1/101	0.000001	71.72	69.88	68.39	61.33	58.66
	200	1/201	0.000001	71.95	69.87	68.17	61.50	59.49
$x^{k+1}=P_\Omega(x^k-\alpha_k(C(x^k)+J_C(x^k)^t(x^k-u^k)))$	0	1.0	0.003248	70.00	70.00	70.00	60.00	60.00
	1	1.0	0.002443	78.93	69.41	61.65	66.95	53.04
où u^k est le p.c.c. associé à $C(x^k)$,	2	1.0	0.002119	84.96	68.98	56.05	71.33	48.66
	3	1.0	0.001717	89.36	68.95	51.67	68.94	51.05
$\alpha_k=\frac{\lambda_k \cdot (C(x^k)\ ,\ x^k-u^k)}{\|C(x^k)+J_C(x^k)^t(x^k-u^k)\|^2}$	4	1.0	0.001683	93.10	68.58	48.31	71.81	48.18
	5	1.0	0.001494	92.11	73.20	44.68	68.06	51.93
	6	1.0	0.001459	95.24	72.63	42.12	70.61	49.38
	7	1.0	0.001313	94.58	76.18	39.23	71.39	48.60
	8	1.0	0.001065	96.94	75.76	37.29	69.89	50.10
$0<\varepsilon_1\le\lambda_k\le 2-\varepsilon_2$,	9	1.0	0.001062	99.01	75.24	35.74	71.61	48.38
	10	1.0	0.000997	98.19	77.88	33.91	69.26	50.73
$\varepsilon_2 > 0$	100	1.0	0.000122	115.14	87.16	7.68	70.01	49.98
	200	1.0	0.000053	117.60	88.56	3.83	69.97	50.02

Tableau 7.5.4 (suite)

Type d'algorithme	Iter.No k	α_k	λ_k	Arc A1	Arc A2	Arc A3	Arc B1	Arc B2
$x^{k+1}=P_\Omega(x^k-\alpha_k(C(x^k)+J_C(x^k)^t(x^k-u^k)))$	0	1.999999	0.006497	70.00	70.00	70.00	60.00	60.00
où u^k est le p.c.c. associé à $C(x^k)$,	1	1.999999	0.003853	87.86	68.83	53.30	73.90	46.09
	2	1.999999	0.003773	86.07	79.57	44.35	65.03	54.96
	3	1.999999	0.002645	94.58	77.40	38.00	72.14	47.85
$\alpha_k=\frac{\lambda_k\cdot(C(x^k)\ ,\ x^k-u^k)}{\|C(x^k)+J_C(x^k)^t(x^k-u^k)\|^2}$,	4	1.999999	0.001929	99.31	76.47	34.21	69.05	50.94
	5	1.999999	0.002090	102.95	75.44	31.60	72.23	47.76
	6	1.999999	0.001821	101.11	80.57	28.30	67.51	52.48
$0<\varepsilon_1\le\lambda_k\le 2-\varepsilon_2$,	7	1.999999	0.001263	104.46	79.31	26.22	70.65	49.34
	8	1.999999	0.001533	103.29	82.22	24.47	67.89	52.10
$\varepsilon_2>0$.	9	1.999999	0.001100	106.00	81.06	22.92	70.51	49.48
	10	1.999999	0.001251	107.64	80.44	21.91	69.27	50.72
	100	1.999999	0.000106	117.82	88.84	3.32	69.95	50.04
	200	1.999999	0.000036	119.22	89.55	1.21	69.96	50.03

7.6 Généralisation des méthodes projection pour résoudre un problème d'équilibre général: cas du flot sur les chemins

Dans cette section, nous allons suivre l'exemple de Bertsekas et Gafni (1980) qui effectuent une transformation de variables afin de pouvoir considérer directement les variables de flots sur les chemins. Nous reprendrons les résultats obtenus à la section 4 après cette transformation.

A moins d'avis contraire, les variables utilisées ci-après désignent le flot sur les chemins. Posons

$\bar{C}(x) = A^t C(Ax)$ la fonction de coût sur les chemins, où A est la matrice d'incidence arc-chemin.

On peut noter que $J_{\bar{C}}(x)$, désignant le Jacobien de $\bar{C}$ par rapport à x est égale à $A^t J_C(Ax)A$, où Ax est le flot sur les arcs associé à x.

Mentionnons aussi que Ω^*, l'ensemble des solutions de l'inégalité variationnelle

$$x^* \in \Omega, \ (A^t C(Ax^*), x-x^*) \geq 0 \qquad \forall \ x \in \Omega \tag{7.6.1}$$

est polyédrique, car $\Omega^* = \{x \in \Omega \mid Ax = z^*\}$ où z^* est l'unique flot sur les arcs à l'équilibre. Nous allons maintenant écrire la condition nécessaire et suffisante d'optimalité pour le calcul d'un flot d'équilibre. Les fonctions de coût sur les arcs sont supposées strictement monotones.

Proposition 7.6.1

Soit F, G et H, trois matrices symétriques et définies positives, ρ_1, ρ_2 et ρ_3 trois nombres réels strictement positifs, alors

$$P_{\Omega}\left[x^*-\rho_1 F^{-1}(\bar{C}(x^*)+\rho_2 G^{-1} J_{\bar{C}}(x^*)^t(x^*-y^*))\right] = x^* \qquad (7.6.2)$$

$$P_{\Omega}\left[y^*-\rho_3 H^{-1}\bar{C}(x^*)\right] = y^* \qquad (7.6.3)$$

si et seulement si

$Ax^* = Ay^* = z^*$ est l'unique flot sur les arcs à l'équilibre.

Preuve

Très similaire à celle de la proposition 7.4.1. □

L'algorithme de la section 4 peut maintenant s'écrire: (Ω représente l'ensemble réalisable des flots sur les chemins)

Etape 0. Soit ρ_1, ρ_2, $\rho_3 > 0$; soit les matrices F, G, H symétriques et définies positives,
soit x^0, $y^0 \in \Omega$ donnés.
Posons $k \leftarrow 0$.

Etape 1. Calculer

$$x^{k+1} = P_{\Omega}[x^k-\rho_1 F^{-1}(\bar{C}(x^k)+\rho_2 G^{-1} J_{\bar{C}}(x^k)^t(x^k-y^k))] \qquad (7.6.4)$$

$$y^{k+1} = P_{\Omega}[y^k-\rho_3 H^{-1}\bar{C}(x^k)] \qquad (7.6.5)$$

Poser $k \leftarrow k+1$. Aller à l'étape 1.

L'algorithme s'arrête lorsque $x^{k+1} = x^k$ et $y^{k+1} = y^k$.

En supposant la forte monotonie des fonctions de coût sur les arcs, nous voulons montrer que cet algorithme génère un point fixe. La condition de Lipschitz des fonctions de coût sur les arcs n'a pas été utilisée ici.

Proposition 7.6.2

Soit $\bar{\gamma}$ le maximum sur Ω de la valeur propre maximale de la matrice symétrique définie positive: $[J_C(Ax)]\ AF^{-1}\ A^t\ [J_C(Ax)]^t$, (7.6.6)

en posant $\rho_1 = \rho_3$, $F = H$ et $G = I$ la matrice identité, deux possibilités peuvent se présenter:

a) $Ax^i \neq Ay^i \quad \forall\ i \geq 0$: en choisissant ρ_1 et ρ_2 suffisamment petit,

$$\lim_{k\to\infty} (\bar{C}(x^k),\ x^k - y^k) = 0 \quad \text{et,}$$

$$\|x^{k+1} - y^{k+1}\|_F \leq \lambda \|x^k - y^k\|_F,\ k \geq 0 \quad \text{où}$$

$\lambda = \{1-(1/\mu)(2\alpha\xi - \rho_1\rho_2\bar{\gamma}M)\rho_1\rho_2\}^{\frac{1}{2}}$ est une constante $\in (0,1)$.

b) $\exists i \geq 0$ telle que $Ax^i = Ay^i$: notre approche est celle de Bertsekas et Gafni (1980).

Preuve

1er cas: $Ax^i \neq Ay^i \quad \forall\ i \geq 0$.

En posant $C = \bar{C}$ dans la relation (7.4.12), on obtient:

$$\|x^{k+1} - y^{k+1}\|_F^2 \leq \|x^k - y^k\|_F^2 + \rho_1^2\rho_2^2 (x^k - y^k,\ J_{\bar{C}}(x^k) F^{-1} J_{\bar{C}}(x^k)^t (x^k - y^k)) \qquad (7.6.7)$$

$$-2\rho_1\rho_2 (x^k - y^k,\ J_{\bar{C}}(x^k)^t (x^k - y^k))$$

Or,

$$(x^k-y^k,\ J_{\bar{C}}(x^k)F^{-1}J_{\bar{C}}(x^k)^t(x^k-y^k)) = (Ax^k-Ay^k,\ J_C(Ax^k)AF^{-1}A^tJ_C^t(Ax^k)\ (Ax^k-Ay^k))$$

$$\leq \bar{\gamma}\,(Ax^k-Ay^k, Ax^k-Ay^k) \quad \text{d'après (7.6.6)}$$

$$\leq \bar{\gamma}\,M(x^k-y^k, x^k-y^k) \quad \text{où } M = \|A\|$$

(voir Luenberger, 1969)*

De plus,

$$(x^k-y^k,\ J_{\bar{C}}^t(x^k)(x^k-y^k)) = (Ax^k-Ay^k,\ J_C^t(Ax^k)(Ax^k-Ay^k))$$

$$\geq \alpha(Ax^k-Ay^k, Ax^k-Ay^k) \quad \text{où } \alpha>0 \text{ est le paramètre de forte monotonie}$$

$$\geq \alpha\,\xi\,(x^k-y^k, x^k-y^k) \quad \text{où } \xi \text{ est la plus petite valeur propre non-nulle de la matrice symétrique semi-définie positive } A^tA.$$

Par conséquent, la relation (7.6.7) peut s'écrire:

$$\|x^{k+1}-y^{k+1}\|_F^2 \leq \|x^k-y^k\|_F^2 + (\rho_1^2\rho_2^2\bar{\gamma}M - 2\rho_1\rho_2\alpha\xi)(x^k-y^k,\ x^k-y^k).$$

Nous choisissons ρ_1 et ρ_2 telles que $\rho_1\rho_2 < 2\alpha\xi/\bar{\gamma}M$. Nous obtenons enfin

$$\|x^{k+1}-y^{k+1}\|_F^2 \leq \{1-(1/\mu)\,\rho_1\rho_2(2\alpha\xi-\rho_1\rho_2\bar{\gamma}M)\}\,\|x^k-y^k\|_F^2 \qquad (7.6.8)$$

La relation (7.6.8) indique la présence d'une contraction relativement à la norme $\|x\|_F$.

D'où $$\lim_{k\to\infty} (C(Ax^k), Ax^k-Ay^k) = 0 \qquad (7.6.9)$$

Enfin, en posant $\lambda = \{1-(1/\mu)\rho_1\rho_2(2\alpha\xi-\rho_1\rho_2\bar{\gamma}M)\}^{\frac{1}{2}}$, la relation suivante est obtenue:

$$\|x^k-y^k\|_F \leq \lambda^k\|x^0-y^0\|_F \qquad (7.6.10)$$

<u>2e cas</u>: $Ax^k = Ay^k$

A partir des relations (7.6.4) et (7.6.5), on obtient:

$$(x^{k+1}-x^k+\rho_1 F^{-1}\bar{C}(x^k), x-x^{k+1}) \geq 0 \qquad \forall\, x \in \Omega$$

$$(y^{k+1}-y^k+\rho_1 F^{-1}\bar{C}(x^k), y-y^{k+1}) \geq 0 \qquad \forall\, y \in \Omega$$

ce qui donne

$$[(x^{k+1}-y^{k+1}) - (x^k-y^k),\ y^{k+1}-x^{k+1}] \geq 0$$

c'est-à-dire

$$[Ax^{k+1}-Ay^{k+1},\ Ay^{k+1}-Ax^{k+1}] \geq 0$$

et enfin,

$$Ax^{k+1} = Ay^{k+1} .$$

Donc, $Ax^{k+m} = Ay^{k+m} \quad \forall\, m \geq 0; \quad \lim_{k\to\infty} (C(Ax^k), Ax^k-Ay^k) = 0.$

Ainsi, selon le résultat de Bertsekas et Gafni (1980), en prenant $\rho_1 > 0$ suffisamment petit, la suite $\{x^k\}$ converge vers un point $x^* \in \Omega^*$, la suite $\{y^k\}$ converge vers un point $y^* \in \Omega^*$. (Voir la section 7.2 pour plus de détails.)

Nous allons maintenant démontrer que l'algorithme précédent génère un point fixe.

Proposition 7.6.3

Supposons que $Ax^k \neq Ay^k \quad \forall\, k \geq 0$, que les conditions de la proposition 7.6.2 sont satisfaites et que la continuité du Jacobien de la fonction de coût est assumée, alors l'algorithme précédent génère un point fixe à la condition que ρ_1 et ρ_2 soient choisis suffisamment petits.

Preuve

Soient $\{x^k\}_{k=0}^{\infty}$, $\{y^k\}_{k=0}^{\infty}$ et $\{\bar{x}^k\}_{k=0}^{\infty}$, $\{\bar{y}^k\}_{k=0}^{\infty}$ des suites de points possiblement générées par cet algorithme. Soit γ le maximum sur Ω de la valeur propre maximale de la matrice symétrique définie positive:

$$[J_C(Ax)]^t\, AF^{-1}A^t\, [J_C(Ax)]. \tag{7.6.11}$$

A partir de la relation (7.4.18), on peut écrire:

$$\begin{aligned}\|x^{k+1}-\bar{x}^{k+1}\|_F^2 \leq \|x^k-\bar{x}^k\|_F^2 &+ \rho_1^2(\bar{C}(x^k)-\bar{C}(x^k),\ F^{-1}(\bar{C}(x^k)-\bar{C}(\bar{x}^k))) \\ &-2\rho_1(\bar{C}(x^k)-\bar{C}(\bar{x}^k),x^k-\bar{x}^k) \qquad (7.6.12)\\ &+\rho_1^2\rho_2^2\|F^{-1}(J_{\bar{C}}(x^k)^t(x^k-y^k)-J_{\bar{C}}(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F^2 \\ &+2\rho_1\rho_2\|F^{-1}(J_{\bar{C}}(x^k)^t(x^k-y^k)-J_{\bar{C}}(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F\cdot\|x^k-\bar{x}^k\|_F \\ &+2\rho_1^2\rho_2\|F^{-1}(J_{\bar{C}}(x^k)^t(x^k-y^k)-J_{\bar{C}}(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F\cdot\|F^{-1}(\bar{C}(x^k)-\bar{C}(\bar{x}^k))\|_F\ .\end{aligned}$$

Calculons maintenant une borne supérieure à la quantité $\|F^{-1}(J_{\bar{C}}(x^k)^t(x^k-y^k)-J_{\bar{C}}(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F$. Nous obtenons

$$\|F^{-1}(J_{\bar{C}}(x^k)^t(x^k-y^k)-J_{\bar{C}}(\bar{x}^k)^t(\bar{x}^k-\bar{y}^k))\|_F$$

$$= \|F^{-1}A^t(J_C(Ax^k)^t(Ax^k-Ay^k)-J_C(A\bar{x}^k)^t(A\bar{x}^k-A\bar{y}^k))\|_F$$

$$\leq \|F^{-1}A^t\|_F \cdot \|J_C(Ax^k)^t(Ax^k-Ay^k)-J_C(A\bar{x}^k)^t(A\bar{x}^k-A\bar{y}^k)\|_F \qquad (7.6.13)$$

$$\leq \bar{a}\,\lambda^k \quad \text{où } \lambda \text{ est défini à la proposition 7.6.2,}$$

et $\bar{a} = \tilde{\gamma}\{\|Ax^0-Ay^0\|_F+\|A\bar{x}^0-A\bar{y}^0\|_F\}\cdot\|F^{-1}A^t\|_F$, $\tilde{\gamma}$ étant définie à la proposition 7.4.3.

Cette quantité décroît strictement vers 0 lorsque k croît.

De plus,

$$\|F^{-1}(\bar{C}(x^k)-\bar{C}(\bar{x}^k))\|_F^2$$

$$= (C(Ax^k)-C(A\bar{x}^k),\ AF^{-1}A^t(C(Ax^k)-C(A\bar{x}^k)))$$

$$\leq (Ax^k-A\bar{x}^k,\ [\tfrac{\partial c(Ax)}{\partial Ax}]^t AF^{-1}A^t[\tfrac{\partial c(ax)}{\partial Ax}](Ax^k-A\bar{x}^k)) \text{ d'après l'appendice 2 de Dafermos (1980).}$$

$$\leq \gamma(Ax^k-A\bar{x}^k, Ax^k-A\bar{x}^k) \qquad \text{d'après (7.6.11)}$$

$$\leq \gamma M(x^k-\bar{x}^k, x^k-\bar{x}^k) \qquad (M=\|A\|, \text{ voir Luenberger 1969})^*$$

$$\leq \frac{\gamma M}{\bar{u}}\,\|x^k-\bar{x}^k\|_F^2 \quad \text{où } \bar{u} \text{ est la valeur propre minimale de F.} \qquad (7.6.14)$$

A partir de (7.6.13), (7.6.14) et du fait que la fonction de coût sur les arcs est fortement monotone de rayon $\alpha > 0$, l'inégalité (7.6.12) est simplifiée comme suit:

---

* $\|A\| = \sqrt{\lambda_{max}}$ où λ_{max} est la plus grande valeur propre de la matrice symétrique et semi-définie positive A^tA.

$$\|x^{k+1}-\bar{x}^{k+1}\|_F^2 \leq \{1-(1/u)\rho_1\rho_2(2\alpha\xi-\gamma\rho_1\rho_2 M)\}\cdot\|x^k-\bar{x}^k\|_F^2$$

$$+ \bar{a}\lambda^k\{\rho_1^2\rho_2^2\bar{a}\lambda^k+2\rho_1\rho_2\|x^k-\bar{x}^k\|_F+2\rho_1^2\rho_2(\tfrac{\gamma M}{u})^{\frac{1}{2}}\|x^k-\bar{x}^k\|_F\} \quad (7.6.15)$$

En posant $\bar{\lambda} = \{1-(\frac{1}{u})\rho_1\rho_2(2\alpha\xi-\gamma\rho_1\rho_2 M)\}$,

$$b = \sup_k \{\rho_1^2\rho_2^2\bar{a}\lambda^k+2\rho_1\rho_2\|x^k-\bar{x}^k\|_F+2\rho_1^2\rho_2(\tfrac{\gamma M}{u})^{\frac{1}{2}}\|x^k-\bar{x}^k\|_F\}$$

et en procédant par induction dans (7.6.15), un choix suffisamment petit de ρ_1 et ρ_2 implique que

$$\|x^k-\bar{x}^k\|_F \xrightarrow[k\to\infty]{} 0 .$$

Ainsi, le résultat de la proposition (7.6.3) s'ensuit. □

On obtient des estimés du taux de convergence très similaires à ceux obtenus à la proposition 7.4.4.

Terminons cette section en affirmant que l'hypothèse $G_k = G \quad \forall\, k \geq 0$ pourrait sûrement être omise. Les résultats de Bertsekas et Gafni (1980) à ce sujet seraient à considérer.

CHAPITRE 8

EXTENSION AU CAS DE LA DEMANDE ELASTIQUE

8.1 INTRODUCTION

Jusqu'à maintenant, nous avons considéré le problème de la recherche d'un flot d'équilibre lorsque la demande est constante. Un modèle plus général serait celui dans lequel la demande est élastique. Nous étudions dans ce chapitre des extensions de la méthode d'accumulation des contraintes pour la résolution d'un problème de trafic d'équilibre avec demande élastique. Nous considérons en détail le cas où l'inverse de la fonction de demande n'est pas connue explicitement.

Soit $H(u): R_+^{n_p} \to R_+^{n_p}$, la fonction de demande qui dépend du vecteur de coût pour chaque paire O/D, où n_p est le nombre de paires O/D, nous supposerons que H est inversible, c'est-à-dire $u = H^{-1}(d)$. Mais, nous ne connaissons pas toujours la forme explicite de l'inverse de H. Le cas spécial où $H(u) = (H_1(u_1), \ldots, H_{n_p}(u_{n_p})$ sera examiné plus en détail.

Posons Ω_d l'ensemble des flots réalisables ayant comme vecteur de demande d, alors le premier principe de Wardrop s'énonce comme suit (Dafermos, 1980):

$$(C(x^*),x-x^*) - (H^{-1}(d^*),d-d^*) \geq 0 \qquad \forall x \in \Omega_d, \forall d. \tag{8.1.1}$$

Nous pouvons maintenant démontrer l'équivalence suivante:

PROPOSITION 8.1.1:

Si $G = \begin{bmatrix} C \\ -H^{-1} \end{bmatrix}$ est un opérateur continu et monotone sur l'ensemble des flots réalisables supposé convexe, alors la relation (8.1.1) est équivalente à

$$(C(x),x-x^*) - (H^{-1}(d),d-d^*) \geq 0 \qquad \forall x \in \Omega_d, \forall d. \tag{8.1.2}$$

DEMONSTRATION: (Voir proposition 1.3.1)

Lorsque le réseau est fortement connecté, lorsque les fonctions de coût sont positives, continues et strictement monotones, lorsque les fonctions de demande sont non-négatives, continues, bornées supérieurement et strictement monotones, alors il existe une solution unique de flot d'équilibre satisfaisant (8.1.1), et par conséquent, (8.1.2) (Aashtiani, 1979).

Remarquons qu'aucune hypothèse de symétrie sur C ou H n'est imposée.

Jusqu'à maintenant, plusieurs méthodes ont été examinées dans le cadre de ce modèle. Une approche de projection a été considérée par Dafermos (1980c); des méthodes de décomposition exigeant des conditions suffisantes de convergence relativement restrictives ont été discutées par Dafermos (1980b) et Florian (1981) et des méthodes d'approximation de g par le premier terme de la série de Taylor (Aashtiani et Magnanti, 1979) ont été étudiées.

De même, Fisk & Nguyen (1980a) suggèrent une méthode d'approximations successives qui consiste à résoudre au moins approximativement un problème d'équilibre avec demandes fixes à chaque itération. Dans ce cadre de travail, la méthode d'accumulation des contraintes pourrait servir à résoudre le problème d'équilibre avec demandes fixes à chaque itération. Un survol détaillé de toutes ces méthodes ainsi que le cadre dans lequel elles s'appliquent a été fait par Fisk et Boyce (1981).

Dans les deux prochaines sections, nous allons développer respectivement une extension de la méthode d'accumulation des contraintes et une adaptation de la méthode de Benders, pour résoudre l'inéquation (8.1.2).

8.2 METHODE D'ACCUMULATION DES CONTRAINTES

Afin de développer notre méthode d'accumulation des contraintes, nous allons réécrire la relation (8.1.2) sous la forme d'un problème MIN-MAX convexe comme suit:

$$\min_{x \in \Omega_d} \max_{y \in \Omega_d} (C(y), x-y) - (H^{-1}(g), d-g) \qquad (8.2.1)$$

$$d \in D \quad g \in D$$

où D est un ensemble convexe, compact et de structure simple contenant chaque vecteur de demande réalisable.

Par exemple, D peut être défini comme ci-après:

$$\{d \in R^{n_p} \mid 0 \le d_i \le \bar{d}_i, \quad i = 1,\ldots,n_p\}.$$

Supposons maintenant que les points $\binom{y^1}{g^1},\ldots,\binom{y^k}{g^k}$ tels que $y^j \in \Omega_{g^j}$, $g^j \in D$, $j = 1,\ldots,k$ sont connus, alors nous considérons le problème approximé (linéaire) de (8.2.1) suivant:

$$\underset{x \in \Omega_d}{\text{Max}} \ \underset{i=1,\ldots,k}{\text{Min}} \ (C(y^i), y^i - x) - (H^{-1}(g^i), g^i - d) \qquad (8.2.2)$$

$$d \in D$$

Nous pouvons maintenant énoncer la méthode d'accumulation des contraintes pour ce problème d'équilibre:

A l'étape k, il s'agit de déterminer une solution optimale (x^k, d^k) de (8.2.2). Puis, en posant x_o^k la valeur de l'objectif de (8.2.2) à l'optimum, alors le critère d'arrêt s'en suit:

"Si $x_o^k \le 0$ alors c'est terminé".

Enfin, le point suivant est généré comme suit:

Nous devons calculer

$$\binom{y^{k+1}}{g^{k+1}} = \binom{y^k}{g^k} + \theta^k \binom{x^k - y^k}{d^k - g^k}$$

où

$$R\left(\binom{y^{k+1}}{g^{k+1}}\right) = \underset{\theta \in [0,1]}{\text{Min}} \ R\left(\binom{y^k}{g^k} + \theta \binom{x^k - y^k}{d^k - g^k}\right) \qquad (8.2.3)$$

et,

$$R\left(\binom{y}{g}\right) = \{(C(y),\ y^k - x^k) - (H^{-1}(g),\ g^k - d^k)\}^2 .$$

Et ainsi de suite.

En se servant du fait que $W = \{\binom{x}{d} \mid x \in \Omega_d,\ d \in D\}$ convexe et compact et en supposant la stricte monotonie de la fonction de coût C et de l'inverse de la fonction de demande $-H^{-1}$, nous pouvons étendre les résultats de convergence de la section (2.2.2). En effet, en remarquant que R est strictement décroissante sur [0,1], nous montrons que la solution $\binom{y^{k+1}}{g^{k+1}}$ permet d'éliminer

la solution courante $\begin{pmatrix} x^k \\ d^k \end{pmatrix}$ du problème linéaire.

La méthode de calcul résultante est très similaire à celle développée précédemment (chapitre 2 et 3). Nous effectuons une transformation de variables qui consiste à exprimer chaque vecteur $\begin{pmatrix} x \\ d \end{pmatrix}$ comme une combinaison linéaire convexe des points extrémaux de W. Nous obtenons de cette façon le problème équivalent à (8.2.2) suivant:

$$\text{Max } x_o \qquad (8.2.4)$$

$$\sum_j \lambda_j = 1 \qquad (\rho)$$

$$x_o + \sum_j (A_i^j - B_i^j)\lambda_j + e_i = c_i,\ i = 1,2,\ldots,k \qquad (\pi)$$

$$\lambda_j \geq 0$$

où

$A_i^j = (C(y^i), \tilde{y}^j)$

$B_i^j = (H^{-1}(g^i), \tilde{g}^j)$

$c_i = (C(y^i), y^i) - (H^{-1}(g^i), g^i)$, $H^{-1}(g^i)$ supposés connus,

e_i = i ième variable d'écart

$(\tilde{y}^j, \tilde{g}^j)$ est un point extrémal de R.

Dès lors, dans l'application de la forme révisée de la méthode du simplexe pour résoudre (8.2.4), les étapes de la suppression des contraintes non-actives, de l'initialisation du tableau du simplexe, le calcul des multiplicateurs de Lagrange, la recherche de la colonne à faire sortir de la base et le pivotage sont identiques à la section 3.2.

Nous allons considérer en détail les deux étapes suivantes:

a) <u>Recherche de la colonne à faire entrer dans la base</u>

Soit (μ,π) les variables duales associées à (8.2.3). Le coût réduit d'une variable d'écart e_i est $-\pi_i$, soit:

$$\pi_s = \min_i \pi_i$$

si $\pi_s < 0$, e_s est candidat à l'entrée de la base et on passe à la phase de calcul suivante.

Dans le cas contraire, nous allons examiner le coût réduit des variables λ_j:

$$\bar{c}_j = -\rho + \pi^t(A^j - B^j)$$
$$= -\rho - \sum_{i=1}^{k} \pi_i((C(x^i), \tilde{x}^j) - (H^{-1}(d^i), \tilde{d}^j)).$$

La plus grande valeur de $\bar{c}_j$ est déterminée par la résolution du sous-problème:

$$\underset{\substack{y\in\Omega_g \\ g\in D}}{\text{Min}} \left(\sum_{i=1}^{k} \pi_i C(x^i), y) - \left(\sum_{i=1}^{k} \pi_i H^{-1}(d^i), g\right) + \rho \equiv F(y^s, g^s) \right. \qquad (8.2.5)$$

qui se ramène à un problème de plus court chemin pour chaque paire O/D. En effet, soit la j ième paire O/D, $\hat{c}_a = \sum_{i=1}^{k} \pi_i\, c_a(x^i)$ le coût associé à l'arc a, L_j = longueur du plus court chemin de la j ième paire O/D où les coûts associés sur les arcs sont $\{\hat{c}_a\}$, l_j = le chemin en question, $\hat{d}_j = \sum_{i=1}^{k} \pi_i\, H_j^{-1}(d^i)$, alors, la solution de (8.2.5) s'obtient ainsi:

si $L_k \geq \hat{d}_k$ alors $g_k = 0$, $y_1 = 0$ pour tout chemin l appartenant à la k ième paire O/D,

sinon $g_k = \bar{g}_k$, $y_l = \begin{cases} \bar{g}_k & \text{si } l = l_k \\ 0 & \text{si } l \neq l_k \end{cases}$

Noter que les π_i sont toutes non-négatives; si les coûts de transport sont non-négatifs, il n'existe aucun circuit négatif dans le réseau.

Si l'objectif de (8.2.4) est négatif à l'optimum c'est-à-dire, $F(y^s, g^s) < 0$ alors $A^s - B^s$ est la colonne à faire entrer dans la base et la procédure continue.

b) Calcul du pas θ^k et de $H^{-1}(g^{k+1})$

Lorsque l'inverse de la fonction de demande est connu, nous nous servons du fait que la fonction R soit strictement décroissante sur [0,1]. Dans le cas contraire, nous devons formuler autrement le problème (8.2.3).

Sous des hypothèses de continuité, de faible coercivité et de stricte monotonie de -H, la multi-application -H est biunivoque (Ortega & Rheinboldt, 1970). De cette façon, nous pouvons réécrire le problème (8.2.3) comme suit:

$$\text{Min } ((C(y^k + \theta(x^k-y^k)),y^k-x^k) - \sum_{i=1}^{n_p} (\mu_i^k + \gamma_i(\bar{\mu}_i^k-\mu_i^k))\ (d_i^k-g_i^k))^2$$

sujet à

$$H_i \begin{pmatrix} \mu_1^k + \gamma_1(\bar{\mu}_1^k-\mu_1^k) \\ \mu_2^k + \gamma_2(\bar{\mu}_2^k-\mu_2^k) \\ \vdots \\ \mu_{n_p}^k + \gamma_{n_p}(\bar{\mu}_{n_p}^k-\mu_{n_p}^k) \end{pmatrix} = g_i^k + \theta(d_i^k-g_i^k), \quad i = 1,\ldots,n_p,$$

$$0 \leq \theta \leq 1$$

$$0 \leq \gamma_i \leq 1,\ i = 1,\ldots,n_p$$

où les vecteurs μ et $\bar{\mu}$ sont choisis de telle façon qu'il existe toujours une et une seule solution réalisable à ce problème lorsque θ est fixé dans [0,1].

Par exemple, lorsque la condition suivante est imposée sur H bornée:

$\forall x, y \in D, \forall \lambda \in [0,1]$,

$H(\lambda x + (1-\lambda)\ y) \quad \in \{\theta H(x) + (1-\theta)H(y) \mid \theta \in [0,1]\}$,

nous posons $\mu^k = H^{-1}(g^k)$, $\bar{\mu}^k = H^{-1}(d^k)$ qui sont supposés déjà connus.

Puisque le cas où $\theta^k = 1$ est facile à détecter, nous allons supposer $\theta^k \in (0,1)$. Par conséquent, nous sommes amenés à résoudre le système d'équations suivant:

$$(C(y^k + \theta(x^k - y^k)),\ y^k - x^k) - \sum_{i=1}^{n_p} (\mu_i^k + \gamma_i(\bar{\mu}_i^k - \mu_i^k))(d_i^k - g_i^k) = 0 \qquad (8.2.6)$$

$$H_i((\mu_i^k + \gamma_i(\bar{\mu}_i^k - \mu_i^k))_{i=1,2,\ldots,n_p}) = g_i^k + \theta(d_i^k - g_i^k),\ i = 1,\ldots,n_p$$

$$0 \leq \theta \leq 1$$

$$0 \leq \gamma_i \leq 1, \quad i = 1,\ldots,n_p.$$

On peut alors utiliser la méthode de Newton-Raphson généralisée. On fait intervenir la matrice jacobienne d'ordre $n_p + 1$ du système précédent qui a toutefois une structure bien particulière (Dupuis, 1980).

Lorsque le nombre de paires O/D devient grand, le système (8.2.6) est difficile à résoudre. Dans la prochaine section, nous allons développer un nouvel algorithme permettant de réduire la complexité des calculs et l'espace-mémoire exigé.

8.3 ADAPTATION DE LA METHODE DE BENDERS GENERALISEE

La difficulté dans la résolution du problème d'affectation de trafic avec demande variable réside dans la difficulté de calculer l'inverse de la fonction de demande. Pour pallier à cet inconvénient, nous avons formulé à la section précédente le problème de la détermination du pas à chaque itération de la méthode d'accumulation des contraintes dans l'espace des coûts généralisés (le coût des plus courts chemins pour chaque paire O/D). Afin de diminuer la quantité des calculs et de réduire l'espace-mémoire utilisé, nous avons effectué la même transformation de variables que précédemment, mais cette fois, sur le problème de programmation linéaire à résoudre à chaque itération de la méthode d'accumulation des contraintes.

En effet, sous les mêmes hypothèses que précédemment nous assurant que -H est biunivoque, le problème MAX-MIN (linéaire) (8.2.2) peut s'écrire:

$$\text{Max } x_o \qquad (8.3.1)$$

sujet à

$$\begin{bmatrix} x_o \\ \vdots \\ x_o \end{bmatrix} + Ax^1 + \ldots + Ax^{n_p} - B\,H(\mu) \leq c$$

$$\sum_{l \in \phi_i} x_l^i = H_i(\mu), \quad i = 1,\ldots,n_p$$

$$x_l^i \geq 0 \quad \forall l \in \phi_i, \; i = 1,\ldots,n_p$$

$$0 \leq \mu_i \leq \bar{\mu}_i \quad (\bar{\mu}_i \text{ étant fixé}), \; i = 1,\ldots,n_p,$$

où

$$A^t = (C(y^1) \mid C(y^2) \mid \ldots \mid C(y^k)),$$

$$B^t = (H^{-1}(g^1) \mid \ldots \mid H^{-1}(g^k)),$$

$$c^t = (c_1,\ldots,c_k) \quad \text{où } c_i = (C(y^i),y^i) - (H^{-1}(g^i),g^i),$$

ϕ_i = l'ensemble des chemins de la i ième paire O/D.

x_l^i = le flot sur le chemin l de la i ième paire O/D.

$\{H^{-1}(d) \mid d \in D\} \subseteq \{\mu \mid 0 \leq \mu \leq \bar{\mu}\}$.

Nous allons élaborer une version de l'algorithme de Benders (Geoffrion, 1972) adaptée à notre problème. Notons que cette méthode a déjà été appliquée au problème de Beckmann avec demandes variables par Florian et Nguyen (1974) et Gallant (1979).

Nous remarquons que lorsque le vecteur μ est fixé, le problème (8.3.1) se réduit à un programme linéaire.

Ainsi, en projetant le problème (8.3.1) sur l'axe μ, nous obtenons un problème équivalent, mais plus facile à résoudre:

$$\max\ v(\mu)$$

sujet à

$$0 \le \mu \le \bar{\mu}$$

où

$$v(\mu) = \max\ x_o \qquad (8.3.2)$$

sujet à

$$\begin{bmatrix} x_o \\ \vdots \\ x_o \end{bmatrix} + Ax^1 + \ldots + Ax^{n_p} \le c + BH(\mu) \qquad (\delta)$$

$$\sum_{1 \in \phi_i} x_1^i = H_i(\mu),\ i = 1,\ldots,n_p \qquad (\gamma)$$

$$x_1^i \ge 0 \quad \forall 1 \in \phi_i,\ i = 1,\ldots,n_p.$$

Explicitons maintenant l'algorithme de Benders généralisé.

ALGORITHME 8.3.1:

Etape 1. Choisir $\hat{\mu}^o$. Poser J = 1.

Etape 2. Poser $\bar{\mu} = \hat{\mu}^{J-1}$.

Résoudre le problème de programmation linéaire précédent qui consiste à évaluer $v(\bar{\mu})$.

Démotons par $\delta^J_{1,\ldots,k}$ et $\gamma^J_{i=1,\ldots,n_p}$ où

$$\gamma_i^j = \min_{1 \in \phi_i} \sum_{j=1}^{k} \delta_j^J C_1(x^j),$$

Les variables duales respectives de ce problème.

Etape 3. Soit $L^*(\mu,\delta,\gamma) = \min \{-x_o + \delta^t(-c-BH(\mu) + x_o + Ax^1 + \ldots + Ax^{n_p})$

$$+ \delta^t(H(\mu) - (\sum_{1 \in \phi_i} x_1^i)_{i=1,\ldots,n_p})\}$$

sujet à

$$x_1^i \ge 0,\ \forall 1 \in \phi_i,\ i=1,\ldots,n_p.$$

Puisque $\sum_{i=1}^{k} \delta_i = 1$ d'après les conditions de Kuhn-Tucker de (8.3.2) on a donc:

$$L^*(\mu,\delta,\gamma) = \text{Min } \delta^t(Ax^1 + \ldots + Ax^{n_p}) - \gamma^t\Big(\sum_{1\in\phi_i} x_1^i\Big)_{i=1,\ldots,n_p}$$

$$+ N(\mu) + \text{constante}$$

sujet à

$$x_1^i \geq 0,\ \forall 1\in\phi_i,\ i = 1,\ldots,n_p,$$

où

$$N(\mu) = -\delta^t BH(\mu) + \gamma^t H(\mu)$$

$$\text{constante} = -\delta^t c,$$

ce qui s'écrit encore,

$$L^*(\mu,\delta,\gamma) = \text{Min} \sum_{i=1}^{n_p} \sum_{1\in\phi_i} \Big(\sum_{j=1}^{k} \delta_j\, C_1(x^j) - \gamma_1\Big)\, y_1 + N(\mu)$$

$$y_1^i \geq 0,\ \forall 1\in\phi_i,\ i = 1,\ldots,n_p$$

Or, d'après les conditions de Kuhn-Tucker de (8.3.2), nous avons

$$\gamma_i = \min_{1\in\phi_i} \sum_{j=1}^{k} \delta_j C_1(x^j),\ i = 1,\ldots,n_p$$

de sorte que,

$$L^*(\mu,\delta,\gamma) = \gamma^t H(\mu) - \delta^t BH(\mu) - \delta^t c$$

$$= \sum_{i=1}^{n_p} \Big(\min_{1\in\phi_i} \sum_{j=1}^{k} \delta_j C_1(y^j) - \sum_{j=1}^{k} \delta_j H_i^{-1}(g^j)\Big)\, H_i(\mu)$$

$$- \sum_{j=1}^{k} \delta_j\big((C(y^j),y^j) - (H^{-1}(g^j),g^j)\big)$$

D'après les conditions de Kuhn-Tucker de (8.3.1), $\gamma = B^t\delta$ c'est-à-dire,

$$\sum_{j=1}^{k} \delta_j H_i^{-1}(g^j) = \min_{1\in\phi_i} \sum_{j=1}^{k} \delta_j C_1(x^j)$$

de sorte que $\partial L^*(\mu,\delta,\gamma)/\partial\mu = 0$ lorsqu'on est à l'optimum.

Supposons maintenant que nous ne sommes pas à l'optimum de (8.3.1), c'est-à-dire qu'il existe au moins un élément γ_i et un élément $(B^t\delta)_i$ de γ et $B^t\delta$ respectivement qui sont distincts. Puisque H est strictement monotone, il s'en suit d'après Auslender (1976) que

$$\left.\frac{\partial L^*(\mu,\delta,\gamma)^t}{\partial\mu}\right/_{\mu = B^t\delta} (\gamma - B^t\delta) < 0.$$

Par conséquent, nous avons à résoudre à cette étape le problème (8.3.3) suivant:

$$\underset{0\le\mu\le\bar{\mu}}{\text{Min}} \ \underset{j=1,\ldots,J}{\text{Max}} \ \sum_{i=1}^{n_p} \left(\text{Min}_{1\in\phi_i} \sum_{m=1}^{k} \delta_m^j C_1(x^m) - \sum_{m=1}^{k} \delta_m^j H_i^{-1}(g^m)\right) H_i(\mu) - \sum_{m=1}^{k} \delta_m^j c_m.$$

Soit $\hat{\mu}^j$ la solution optimale obtenue, si la précision est suffisante, c'est terminé; nous avons obtenu une solution (approximative) (x_d^k) de (8.2.2) où $H^{-1}(d^k)$ est connu. Lorsque cela ne suffit pas, il faut poser $J \leftarrow J+1$ et retourner à l'étape 2.

Considérons de nouveau le problème (8.3.3). Celui-ci peut s'écrire sous la forme

$$\underset{0\le\mu\le\bar{\mu}}{\text{Min}} \ \underset{j=1,\ldots,J}{\text{Max}} \ \sum_{i=1}^{n_p} a_i^j H_i(\mu) - \eta^j \qquad (8.3.4)$$

où les a_i^j et les η^j sont connus. De façon générale, la résolution de (8.3.4) n'est pas aisée. Par contre, lorsque H_i est affine $\forall i=1,\ldots,n_p$ par exemple, nous pouvons utiliser une approche de sous-gradient pour résoudre (8.3.4).

Etant donné la difficulté concernant la résolution du problème (8.3.4), nous proposons plutôt l'heuristique qui suit:

cela consiste à remplacer $\hat{\mu}^j$ la solution optimale de (8.3.3) par l'optimum $B^t\delta + \hat{\theta}\,(\gamma - B^t\delta)$ de

$$\hat{\theta} = \underset{\theta\in[0,1]}{\text{Arg Min}} \ L^*(B^t\delta + \theta(\gamma - B^t\delta),\delta,\gamma).$$

Cette suggestion provient du fait que $\gamma - B^t\delta$ est une direction de descente de L^* au point $\mu = B^t\delta$.

Nous discuterons maintenant de la génération de la coupe suivante à l'itération k+1. Une approche consiste à se servir uniquement de la solution $\binom{x^k}{d^k}$ de (8.2.2) où $H^{-1}(d^k)$ est connu, pour générer cette coupe. D'autres alternatives sont possibles; toutefois, nous devons tenir compte de la complexité des calculs.

CHAPITRE 9

RESOLUTION D'UN PROBLEME GENERAL DE PROGRAMMATION CONVEXE GRACE A UNE FORMULATION PRIMALE-DUALE

L'importance en mathématiques appliquées d'examiner le problème de minimiser ou de maximiser une certaine fonction f sur un ensemble X ne date pas d'hier. Lorsque la propriété de convexité est présente, cette étude est facilitée. En effet, plusieurs résultats sont obtenus caractérisant l'optimum de ces problèmes. Nous bénéficions alors avantageusement du principe de dualité et des notions qui s'y rattachent.

Dans ce chapitre, nous allons considérer une approche duale pour résoudre notre problème de programmation convexe. Celle-ci offre habituellement une plus grande flexibilité quant au choix des méthodes de révision des variables duales (méthode des sous-gradients (Polyak, 67; Polyak, 69; Polyak, 70; Auslender, 76), méthode d'approximation tangentielle (Hartley et al., 63; Huard, 71). Toutefois, elle a le désavantage de ne pas donner directement une solution primale réalisable. Par conséquent, le but poursuivi dans la prochaine section est d'obtenir une formulation duale exprimée à partir des variables primales seulement.

A l'aide de cette nouvelle formulation, nous allons donner différents algorithmes pour résoudre le problème original (primal). La méthode d'accumulation des contraintes développée dans le contexte des problèmes d'affectation a été étendue à la résolution d'un problème de programmation convexe général.

9.1 UNE FORMULATION PRIMALE-DUALE D'UN PROGRAMME CONVEXE

Soit le problème général suivant:

$$\min_{x \in X} f(x) \qquad (9.1.1)$$

où f est convexe, continûment différentiable sur l'ensemble X non-vide, convexe et compact.

Nous verrons ultérieurement les effets entraînés par une relaxation des hypothèses de convexité et de différentiabilité de f sur X.

Posons

$$G(x) \triangleq \max_{y \in X} (\nabla f(x), x-y) \qquad (9.1.2)$$

et

$$g(y) = \min_{x \in X} (\nabla f(x), x-y) \qquad (9.1.3)$$

En général, la fonction G(x) ne possède aucune propriété de convexité et/ou de différentiabilité sur X. Par contre, la fonction g(y) est concave mais non-nécessairement différentiable sur X.

Soit $\chi \subseteq X$ l'ensemble non-vide des solutions optimales de (9.1.1), $\overline{\chi} \subseteq X$ l'ensemble-solution du problème

$$\min_{x \in X} G(x)$$

alors, nous avons les résultats suivants.

LEMME 9.1.1

1) $G(x) \geq 0$ pour tout $x \in X$
2) $\chi = \overline{\chi}$
3) $G(x) = 0 \Leftrightarrow x \in \chi$

PREUVE

1) évident
2) d'après (Rockafellar, 1972).
3) . Supposons $x \in \chi$, cela entraîne que $G(x) \leq 0$ c'est-à-dire $G(x) = 0$, car χ est aussi l'ensemble-solution de $(\nabla f(x), y-x) \geq 0 \ \forall y \in X$.
   . Supposons $x \in X$ tel que $G(x) = 0$, alors par définition, $(\nabla f(x), y-x) \geq 0$ $\forall y \in X$ de sorte que $x \in \chi$. □

Ce lemme a déjà été énoncé par Hearn (82) lorsque X est un polytope non-vide. L'hypothèse de convexité peut être relaxée comme suit.

PROPOSITION 9.1.1:

Soit f pseudo-convexe et continûment différentiable sur X, alors les relations 1), 2) et 3) du lemme 9.1.1 sont encore satisfaites.

PREUVE

Nous montrerons seulement que $X = \overline{Y}$, les relations 1) et 3) étant évidentes à partir des faits précédents.

$x \in \overline{X} \rightarrow x \in X$ découle du fait que f est pseudo-convexe sur X.

$x \in X \rightarrow x \in \overline{X}$ car $x \in \overline{X}$ est une condition nécessaire d'optimalité du problème (9.1.1). □

Une façon de résoudre le problème (9.1.1) consiste donc à rechercher une solution optimale du programme mathématique

$$\text{Min } G(x)$$
$$\text{sujet à: } x \in X \tag{9.1.4}$$

Or, la fonction G(x) n'est généralement pas différentiable partout de sorte que les méthodes de directions réalisables ne sont pas applicables. Par contre, lorsque G est une fonction convexe sur X, Hearn (82) suggère un algorithme de sous-gradient de Polyak (69) qui permet de minimiser une fonction convexe non-différentiable lorsque la valeur de l'objectif à l'optimum est connue. La méthode de Polyak s'écrit ainsi:

$$x^{k+1} = P_X(x^k - \rho_k(\nabla f(x^k) + \nabla^2 f(x^k)^t(x^k - y^k))), \tag{9.1.5}$$

où

$$\rho_k = \frac{\lambda_k G(x^k)}{\|\nabla f(x^k) + \nabla^2 f(x^k)^t(x^k - y^k)\|^2}$$

$$0 \leq \varepsilon_1 < \lambda_k \leq 2 - \varepsilon_2, \quad \varepsilon_2 > 0,$$

$y^k \in \{z^k \in X \mid z^k$ est solution du problème

$$\text{Max}_{y \in X} \ (\nabla f(x^k), x^k - y) \qquad (9.1.7)$$

$\nabla^2 f$ est la matrice hessienne de f,

$P_X(x)$ indique la projection de x sur X.

Plusieurs variantes de cette méthode ont été données par Bakushinskii et al. (1974).

Une méthode de plan de coupe convexe introduite par Cheney et Goldstein (1959), Kelley (1960) adaptée au problème (9.1.4) pourrait s'énoncer comme suit:

"Trouver x^{k+1} une solution optimale du problème

$$\text{Min}_{x \in X} \ \text{Max}_{i=1,\ldots,k} \ (\nabla f(x^i) + \nabla^2 f(x^i)(x^i - y^i), x - x^i)$$

où y^i est tel que défini par (9.1.7):"

Toutefois, la projection sur l'ensemble X et/ou l'évaluation du Hessien de f au point x^k peuvent s'avérer difficiles. Cela suppose aussi que l'on peut évaluer G de façon exacte; c'est-à-dire, l'ensemble X doit être relativement simple. Enfin, l'hypothèse de convexité de G sur X n'est pas vérifiée en général. Considérons par exemple la situation suivante:

soit $f(x) = e^x, \quad x \in [-10,10] \subseteq R^1$

$\nabla f(x)\ (x-y) = e^x(x-y)$

et $\dfrac{d^2(e^x \cdot (x-y))}{dx^2} = e^x[x-y+2] < 0 \quad$ si $x = 1,\ y = 4.$

Notre objectif ici n'étant pas d'énumérer de façon exhaustive toutes les méthodes permettant de résoudre (9.1.4) dans chaque cas particulier, nous nous en tiendrons à un algorithme de décomposition de type "simplicial", développé par Lawphongpanich et Hearn (1982) pour déterminer un flot d'équilibre dans un réseau routier.

Sachant que l'ensemble X supposé polyédrique peut être représenté par l'enveloppe convexe d'un nombre fini de points extrêmes que l'on connaît implicitement, cet algorithme de décomposition se présente comme suit.

Supposons connus explicitement un sous-ensemble de points extrémaux de X, dénotés $z^1,\ldots,z^k$, il s'agit alors de trouver une solution ϵ_k-optimale du problème approximatif de (9.1.4) qui s'écrit ainsi:

$$\begin{array}{lll} \underset{\substack{\sum_{i=1}^{k} \lambda_i = 1}}{\text{Min}} & \underset{\substack{\sum_{i=1}^{k} \theta_i = 1}}{\text{Max}} & (\nabla f(\sum_{i=1}^{k} \lambda_i z^i),\ \sum_{i=1}^{k} (\lambda_i - \theta_i) z^i) \\ \lambda \geq 0 & \theta \geq 0 & \end{array} \qquad (9.1.8)$$

Soit $\bar{\lambda} = (\bar{\lambda}_1,\ldots,\bar{\lambda}_k)$ une telle solution et $\bar{x} = \sum_{i=1}^{k} \bar{\lambda}_i z^i$, une façon de générer un autre point extrémal consiste à résoudre

$$\underset{y \in X}{\text{Min}}\ (\nabla f(\bar{x}), y)$$

c'est-à-dire, à évaluer $G(\bar{x})$.

Lawphongpanich et Hearn donnent de plus un schéma d'élimination de points extrémaux devenus inutiles.

L'avantage de cette méthode est dans la simplification de l'ensemble X (voir 9.1.8). Nous pouvons utiliser des méthodes de projection très efficaces pour résoudre (9.1.8) (Bertsekas, et Gafni, 80).

Nous allons maintenant examiner les propriétés de la fonction g définie en (9.1.3).

$$\text{Posons } T(y) = \{\bar{x} \in X \mid \bar{x} = \underset{x \in X}{\arg\,\text{Min}}\ (\nabla f(x), x-y)\}, \qquad y \in X.$$

PROPOSITION 9.1.2

Si T(y) est un sous-ensemble convexe de X, $\forall y \in X$, et que f est continuement différentiable, alors chaque solution de

$$\text{Max}_{x \in X}\ g(x) \qquad (9.1.9)$$

est aussi solution de

$$\text{Min}_{x \in X}\ G(x)$$

(Note: $\bar{\bar{X}}$ = l'ensemble-solution de (9.1.10),
$\tilde{X}$ = l'ensemble-solution de (9.1.9))

PREUVE

Soit $\bar{x} \in X$ solution optimale de (9.1.9). Il s'agit de montrer que $g(\bar{x}) = 0$.

Puisque ∇f est continu et X compact, cela entraîne que T est semi-continue supérieurement (s.c.s.).[1] En considérant le théorème de point fixe de Kakutani (1941) qui assure l'existence d'un point fixe d'une multi-application s.c.s. définie sur un ensemble compact Y à des sous-ensembles convexes et fermés de Y, cela nous permet d'affirmer qu'il existe un $\tilde{x} \in X$ tel que $\tilde{x} \in T(\tilde{x})$, par hypothèse.

D'où, $g(\tilde{x}) = 0$, par définition de g.

Mais, $\bar{x}$ est une solution optimale de (9.1.9); par conséquent, nous obtenons:

$$0 \geq g(\bar{x}) \geq g(\tilde{x}) = 0$$

c'est-à-dire, $g(\bar{x}) = 0$.

Ainsi, $(\nabla f(x),\ x - \bar{x}) \geq 0$ pour tout $x \in X$ (9.1.11)

1. Une multi-application $B: X \to Y$ est s.c.s. si B(x) est compact $\forall x \in X$, et pour chaque sous-ensemble ouvert G de Y, l'ensemble $B^{+}G$ est ouvert (Berge, 1963).

Posons $x(\lambda) = \lambda x + (1-\lambda)\bar{x}$ $\forall x \in X$, $\lambda \in (0,1)$. Nous avons alors de (9.1.11)

$$(\nabla f(x(\lambda)), x(\lambda)-\bar{x}) \geq 0 \qquad \text{c'est-à-dire,}$$

$$(\lambda \nabla f(x(\lambda)),\ x-\bar{x}) \geq 0 \tag{9.1.12}$$

En divisant par λ et en laissant tendre λ vers 0 dans la relation (9.1.12), nous obtenons grâce à la continuité de ∇f

$$(\nabla f(\bar{x}), x-\bar{x}) \geq 0 \qquad \text{pour tout } x \in X.$$

Ainsi,

$\bar{x}$ est une solution de (9.1.10). □

Etant donné que l'hypothèse de convexité de l'ensemble $T(y)$, $y \in X$ est difficile à vérifier, nous donnons ici une condition suffisante où cette hypothèse sera toujours vraie.

COROLLAIRE 9.1.2:

Si la fonction $(\nabla f(x), x-\bar{y})$, $\bar{y} \in X$ est quasi-convexe sur X, et le vecteur ∇f est continu, alors l'ensemble-solution du problème

$$\operatorname*{Min}_{x \in X} (\nabla f(x),\ x-\bar{y}) \tag{9.1.13}$$

est convexe, compact, non-vide, et chaque solution de (9.1.9) (et il en existe au moins une!) est aussi solution de (9.1.10).

PREUVE

Puisque la fonction $x \to (\nabla f(x), x-\bar{y})$ est quasi-convexe sur X, on peut montrer facilement que l'ensemble

$$X_{\bar{y}}(\varepsilon) = \{x \in X \mid (\nabla f(x), x-\bar{y}) \leq \varepsilon\} \text{ est convexe, } \bar{y} \in X \text{ fixé.}$$

D'où, en posant $\bar{\varepsilon} = \operatorname*{Min}_{x \in X} (\nabla f(x), x-\bar{y})$, on a que $X_{\bar{y}}(\bar{\varepsilon})$ est convexe, le résultat désiré.

Il reste à montrer que (9.1.9) possède une solution.

Par définition de quasi-convexité, nous avons:

$$\exists x^* \in X,\ (\nabla f(x^*), x-x^*) \geq 0 \quad \forall x \in X$$

$$\Leftrightarrow$$

$$(\nabla f(x), x-x^*) \geq 0 \quad \forall x \in X.$$

Puisque ∇f est continu, cela suit. □

Sous certaines conditions données à la proposition (9.1.2), nous pouvons obtenir au moins une solution au problème (9.1.10) par l'intermédiaire de (9.1.9). Un avantage du problème (9.1.9) est la concavité de la fonction g sur X.

Considérons maintenant 3 exemples qui décrivent assez bien le comportement des problèmes précédents.

Posons $\Omega = \{(x,y,z) \in R^3 \mid x,y,z \geq 0, \quad x+y+z = 2\}$

EXEMPLE 9.1.1

$\min_{x \in \Omega} f_1(X)$.

$$f_1(x,y,z) = x^2 - 1/2y^2 - 1/2z^2 + xy + xz - yz$$

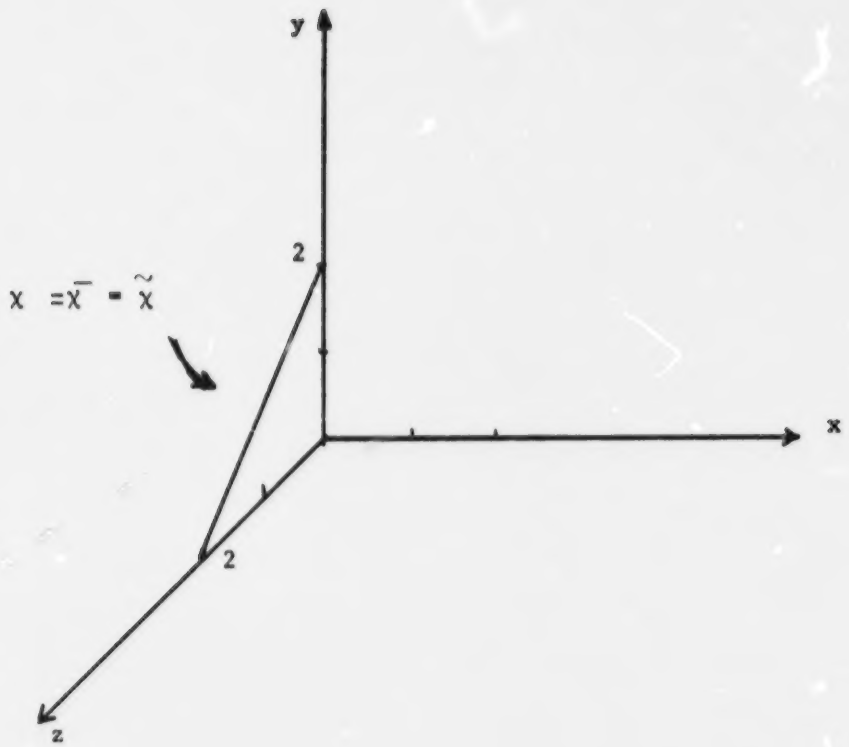
$$\nabla^2 f_1 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

Puisque les valeurs propres de f sont 0, $\sqrt{6}$ et $-\sqrt{6}$, la fonction f_1 n'est pas convexe sur Ω mais elle est quasi-convexe et pseudo-convexe sur Ω.

$$\chi = \overline{\chi} = \widetilde{\chi} = \{(0,\lambda,2-\lambda) \mid \lambda \in [0,2]\}.$$

$g(x) = G(x) = 0$ pour tout $x \in \chi$.

$T(y) = \chi$ pour tout $y \in \Omega$.



Graphique 9.1.1: Représentation de l'ensemble-solution des problèmes associés à l'exemple 9.1.1.

EXEMPLE 9.1.2

Min $f_2(X)$.
$X \in \Omega$

$$f_2(x,y,z) = 1/2x^2 - 1/2y^2 - 1/2z^2 + xy + xz - yz$$

$$\nabla^2 f_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

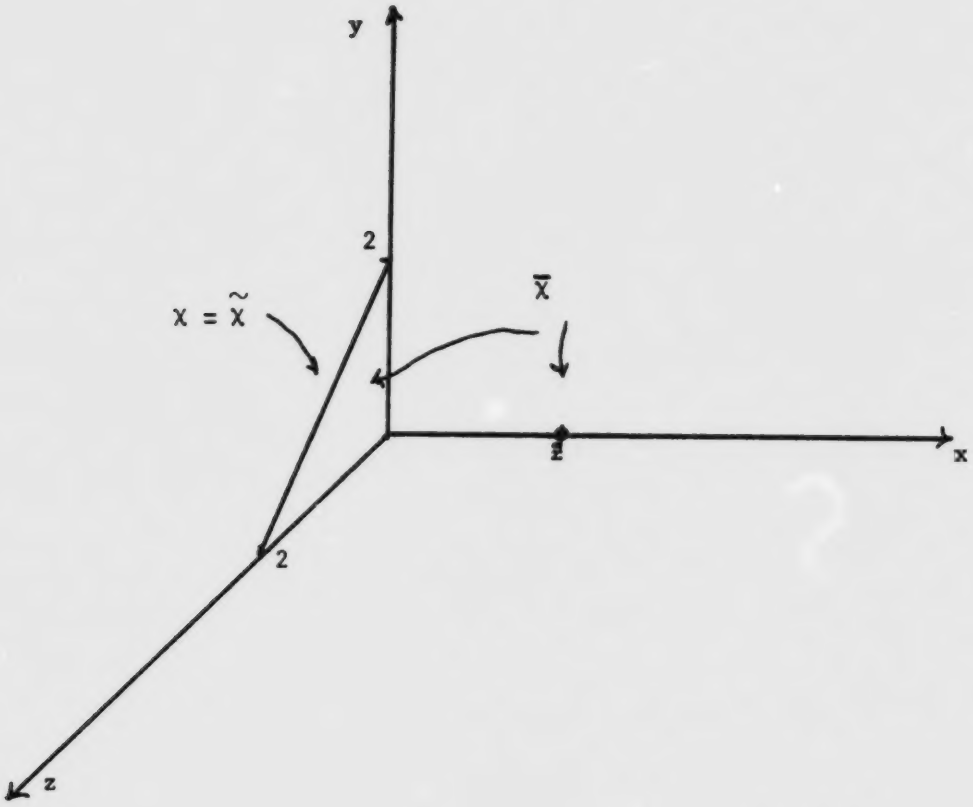
Les valeurs propres de cette matrice sont $0, (1 \mp \sqrt{17})/-2$. Par conséquent, f_2 n'est pas convexe; de plus, elle n'est pas pseudo-convexe mais quasi-convexe sur Ω.

$$\chi = \{(0,\lambda,2-\lambda) \mid \lambda \in [0,2]\}$$

$$\tilde{X} = \{(2,0,0)\} \cup \chi$$

$\tilde{X} = \chi$ et $T(y) = \chi$ pour tout $y \in \Omega$.

$g(x) = 0 = G(y) \quad \forall x \in \tilde{X}, \quad \forall y \in \bar{X}.$



Graphique 9.1.2: Représentation de l'ensemble-solution des problèmes associés à l'exemple 9.1.2

EXEMPLE 9.1.3

$$\text{Soit} \quad C\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} x + y + 2z \\ 2y + z \\ 2y + z \end{bmatrix} ,$$

qui ne peut s'exprimer comme le gradient d'une certaine fonction f, nous allons résoudre les problèmes (9.1.9) et (9.1.10) auxquels on a substitué ∇f par le vecteur C.

Etant donné que les valeurs propres de la partie symétrique du Jacobien de l'opérateur C ne sont pas toutes positives ou nulles, le vecteur C n'est pas monotone sur Ω.

Nous obtenons:

$$\bar{X} = \{(0,\lambda,2-\lambda) \mid \lambda \in [0,1]\} \cup \{(\lambda,1,1-\lambda) \mid \lambda \in [0,1]\}$$

et $G((x,y,z)) = 0$ pour tout $(x,y,z) \in \bar{X}$.

Par contre, il n'existe pas de solution $(\bar{x},\bar{y},\bar{z}) \in \Omega$ à

$$C\begin{pmatrix} x \\ y \\ z \end{pmatrix}^t \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \\ z-\bar{z} \end{pmatrix} \geq 0 \quad \text{pour tout } (x,y,z) \in \Omega.$$

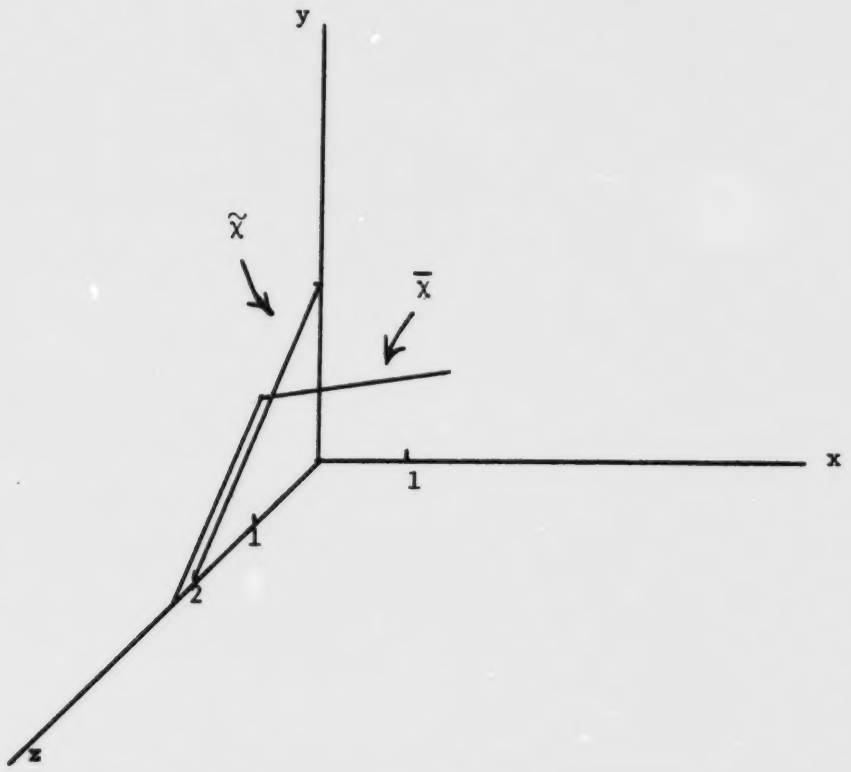
En effet, $\tilde{X} = \{(3-2\sqrt{2},\lambda,-1+2\sqrt{2}-\lambda) \mid \lambda \in [0,-1+2\sqrt{2}]\}$ et

$$g((x,y,z)) = -2(3-2\sqrt{2}) < 0 \quad \text{pour tout } (x,y,z) \in \tilde{X}.$$

Considérons l'unique point de $\tilde{X} \cap \bar{X} = \{(3-2\sqrt{2},1,-2+2\sqrt{2})\}$, nous obtenons

$$T(\tilde{X} \cap \bar{X}) = \{(0,0,2),\ (2-\sqrt{2},\sqrt{2},\ 0)\}$$

qui n'est pas convexe. Les conditions de la proposition 9.1.2 ne sont pas satisfaites.



Graphique 9.1.3: Représentation de l'ensemble-solution des problèmes associés à l'exemple 9.1.3.

Nous verrons maintenant quelques propriétés primales-duales des problèmes (9.1.9) et (9.1.10).

PROPOSITION 9.1.3:

Sous les hypothèses de la proposition (9.1.2), en posant $\tilde{X}$ l'ensemble-solution de (9.1.9), $\bar{X}$ l'ensemble-solution de (9.1.10), les problèmes (9.1.9) et (9.1.10) sont duaux dans le sens suivant:

$$G(x) \geq 0 \geq g(y) \text{ pour tout } x,y \in X, \ \tilde{X} \subseteq \bar{X}$$

et $G(x) = 0 = g(y) \Leftrightarrow x \in \bar{X}, \ y \in \tilde{X}$.

PREUVE

Tous ces résultats découlent de la proposition (9.1.2) à l'exception de celui-ci:

$$G(x) = 0 \Leftrightarrow x \in \bar{X}.$$

Supposons qu'il existe un $x \in X$ tel que $G(x) = 0$. Nous savons que $G(x) \geq 0$ pour tout $x \in X$. Par conséquent, $x \in \bar{X}$ de façon évidente. Considérons maintenant le cas où $x \in \bar{X}$, c'est-à-dire x est une solution optimale de (9.1.10). Supposons que $G(x) > 0$. Cela entraîne que

$$\max_{y \in X} (\nabla f(x), x-y) > 0.$$

Puisque ∇f est continu, alors il existe une solution $\bar{x}$ à l'inégalité variationnelle

$$(\nabla f(\bar{x}), \bar{x}-x) \leq 0 \quad \text{pour tout } x \in X$$

c'est-à-dire,

$$\max_{y \in X} (\nabla f(\bar{x}), \bar{x}-y) \leq 0$$

de sorte que $G(\bar{x}) = 0$ entraînant une contradiction. □

Examinons maintenant les conditions à imposer pour obtenir l'égalité des ensembles $\overline{X}$ et $\widetilde{X}$.

PROPOSITION 9.1.4

En supposant la continuité et la monotonie de ∇f sur X, on obtient:

$$\widetilde{X} = \overline{X}.$$

PREUVE

Nous faisons appel à la propriété de monotonie de ∇f pour montrer que $\widetilde{X} \supseteq \overline{X}$ (voir par exemple Glowinski et al.,1976; Auslender, 1976; Nguyen et Dupuis, 1981). □

Comparons maintenant l'ensemble-solution $\widetilde{X}$ avec celui de (9.1.1).

PROPOSITION 9.1.5

En supposant la quasi-convexité de f sur X et la pseudo-convexité de f sur X, on obtient:

$$X = \widetilde{X} = \overline{X}.$$

PREUVE

Voir la proposition 9.1.1.

Les propositions (9.1.3) - (9.1.5) avaient déjà été démontrées dans le cas simple où f est convexe sur X. (Hearn & Nguyen, 1982).

Une dernière propriété à examiner est la différentiabilité de g. De façon générale, la fonction g n'est pas différentiable partout. Par contre, une condition de stricte convexité entraînant la stricte monotonie de ∇f est suffisante pour s'assurer la différentiabilité de g.

PROPOSITION 9.1.6

Soit la fonction $(\nabla f(x), x-y)$, $y \in X$ fixé supposée strictement convexe sur X et f continuement différentiable sur X, alors g est différentiable sur X et ∇f

est strictement monotone sur X lorsque $\nabla^2 f$ existe.

PREUVE

La stricte convexité de cette fonction entraîne l'unicité de la solution du problème

$$\min_{x \in X} (\nabla f(x), x-y) \qquad \text{pour tout } y \in X,$$

ce qui implique que g est différentiable sur X.

D'après l'inégalité du gradient,

$$(\nabla f(y), y-x) > (\nabla f(\tilde{y}), \tilde{y}-x) + (\nabla f(\tilde{y}) + \nabla^2 f(\tilde{y})^t (\tilde{y}-x),\ y-\tilde{y})$$

pour tout $y \neq \tilde{y} \in X$, pour tout $x \in X$,

c'est-à-dire,

$$(\nabla f(y)-\nabla f(\tilde{y}), y-x) > (y-\tilde{y}, \nabla^2 f(\tilde{y})(\tilde{y}-x))$$

pour tout $y \neq \tilde{y} \in X$, pour tout $x \in X$.

Posons $x = \tilde{y}$; nous obtenons alors

$$(\nabla f(y) - \nabla f(\tilde{y}),\ y-\tilde{y}) > 0 \quad \text{pour tout } y \neq \tilde{y} \in X \qquad \square$$

COROLLAIRE 9.1.6

Soit $X \subset R^p$, $A \in \mathcal{M}_{p\times p}$ et $b \in R^p$, si ∇f est une application affine c'est-à-dire $\nabla f(x) = Ax + b$, alors les 2 conditions suivantes sont équivalentes:

a) la fonction $x \to (\nabla f(x), x-y)$ est strictement convexe sur X où $y \in X$ est fixé.
b) ∇f est strictement monotone sur X.

Avant de terminer cette section, nous allons considérer la fonction $h(x,y) = (\nabla f(x), x-y)$ définie sur $X\times X$. L'intérêt de calculer un point de selle de h sur $X\times X$ semble évident.

Le lemme suivant a été démontré par Hearn et Nguyen (1982).

LEMME 9.1.2:

Si $(\bar{x},\bar{y}) \in X \times X$ est un point de selle de h, alors $h(\bar{x},\bar{y}) = 0$.
(Remarquons qu'aucune hypothèse de convexité sur f est imposée).

Examinons maintenant les liens qui existent entre l'ensemble des points de selle de h sur $X \times X$ et les ensembles-solutions $\mathcal{X}$, $\bar{\mathcal{X}}$ et $\tilde{\mathcal{X}}$. Jusqu'à ce jour, les conditions d'existence et d'unicité des points de selle d'une fonction donnée et les liens entre les solutions optimales du problème primal et celles du dual lagrangien correspondant ont toujours été étudiés dans le cadre de fonctions de selle convexe-concave générales (Mangasarian et al., 1965; Rockafellar, 68). Les résultats qui suivent diffèrent dans ce sens.

PROPOSITION 9.1.7:

Sous les hypothèses de la proposition (9.1.2), $(\bar{x},\bar{y})$ est un point de selle de h sur X×X si et seulement si $\bar{x} \in \bar{\mathcal{X}}$ et $\bar{y} \in \tilde{\mathcal{X}}$.

PREUVE

Si $(\bar{x},\bar{y})$ est un point de selle de h sur X×X, alors c'est évident que $\bar{x} \in \bar{\mathcal{X}}$ et $\bar{y} \in \tilde{\mathcal{X}}$ d'après la définition d'un point de selle de h sur $X \times X$:

$$h(\bar{x},y) \leq h(\bar{x},\bar{y}) \leq h(x,\bar{y}) \quad \text{pour tout } x,y \in X.$$

Inversement, si $\bar{x} \in \bar{\mathcal{X}}$ et $y \in \tilde{\mathcal{X}}$, alors

$$0 = G(\bar{x}) \geq (\nabla f(\bar{x}),\bar{x}-y) \quad \text{pour tout } y \in X$$

et, $$0 = g(\bar{y}) \geq (\nabla f(x), x-\bar{y}) \quad \text{pour tout } x \in X.$$

D'où, $(\nabla f(\bar{x}), \bar{x}-\bar{y}) \leq 0 \leq (\nabla f(\bar{x}), \bar{x}-\bar{y})$ ce qui entraîne $(\nabla f(\bar{x}), \bar{x}-\bar{y}) = 0$. Nous avons donc un point de selle de h sur $X \times X$. □

Les propositions (9.1.3) et (9.1.7) sont encore vraies lorsque les hypothèses de la proposition (9.1.2) sont remplacées par la monotonie et la continuité de ∇f sur l'ensemble X. L'exemple (9.1.3) est un cas simple où les hypothèses de la proposition (9.1.2) ne sont pas satisfaites et où il n'existe pas de point de selle de h sur $X \times X$.

Dans la prochaine section, nous verrons différentes méthodes pour résoudre le problème de programmation convexe énoncé en (9.1.9) et pour calculer un point de selle de la fonction h sur $X \times X$.

9.2 UNE NOUVELLE APPROCHE DE RESOLUTION D'UN PROBLEME DE PROGRAMMATION CONVEXE

Nous avons vu dans la section précédente qu'un problème de programmation convexe se caractérise par une formulation duale ou primale-duale ne faisant intervenir que des variables primales seulement.

Nous aborderons maintenant la résolution de ce problème de programmation convexe en utilisant judicieusement ces nouvelles formulations.

Une première approche consiste à rechercher un point de selle de la fonction $h(x,y) = (\nabla f(x), x-y)$ sur $X \times X$.

Différentes méthodes d'approximations successives ont été examinées (Dem'yanov, Malozemov, 1974), des algorithmes de relaxation (Cohen, 1980), des méthodes de projection (Auslender), et bien d'autres procédés de calcul (Dem'yanov et Pevnyi, 1972).

Chacune de ces méthodes est adéquate pour certaines classes de problèmes; il serait fastidieux et inutile de les expliciter en détail. Nous avons vu dans les chapitres précédents, lesquelles peuvent être appliquées avec succès aux problèmes de transport.

Une seconde approche qui retiendra notre attention d'ici la fin de ce chapitre concerne la résolution de

$$\operatorname{Max}_{x \in X} g(x) \tag{9.2.1}$$

où

$$g(x) = \operatorname{Min}_{y \in X} (\nabla f(y), y-x). \tag{9.2.2}$$

Etant donné que les problèmes (9.2.1) d'intérêt possèdent comme valeur de l'objectif à l'optimum la valeur nulle et que la fonction g est concave sur X, la première idée qui vient à l'esprit consiste à utiliser une approche de

sous-gradient. Malheureusement, le calcul exact d'un sous-gradient de -g est difficile, sinon impossible. Pour remédier à cette situation, Polyak (1971) suggère une méthode de projection en présence de perturbations, comme celles d'approximer les sous-gradients de -g.

Nous pouvons aussi utiliser une méthode de plan de coupe (Kelley, 1960; Zangwill, 1969, pp.296-311) pour résoudre le problème (9.2.1)-(9.2.2). Malheureusement, ces méthodes possèdent le même inconvénient que précédemment à savoir la nécessité de calculer exactement une solution au problème (9.2.2). Parikh (1976) a examiné cette situation. Sa méthode consiste à générer des plans de coupe approximatifs qu'il obtient à l'aide de solutions ε-optimales de (9.2.2) où ε est le degré de précision en vigueur à une étape donnée.

En pratique, la résolution même approchée du problème (9.2.2) à chaque itération d'un algorithme donné peut s'avérer fort coûteuse. L'algorithme que nous préconisons dans cette section, se veut être un palliatif à tous ces soucis.

Nous suggérons une méthode de type des méthodes d'accumulation des contraintes ("cutting-planes methods") qui consiste essentiellement à post-optimiser un problème de programmation linéaire à chaque itération lorsque l'ensemble X est polyédrique.

Cet algorithme a été étudié en détail dans le cadre des problèmes d'équilibre. Nous ne ferons ici qu'énoncer l'algorithme et son résultat de convergence, la preuve de convergence étant basée essentiellement sur les résultats du chapitre 2. La seule différence se situe dans le fait que la fonction de coût C(x) intervenant dans les problèmes d'affectation ne correspond pas de façon générale au gradient d'une fonction donnée. Mais, cela n'intervient d'aucune façon dans les développements mathématiques en question.

Nous supposons dans ce qui suit que f est continuement différentiable sur l'ensemble polyédrique X et qu'il existe une solution à l'inégalité complémentaire suivante (voir section 9.1 pour plus de détails):

$$\bar{x} \in X, \ (\nabla f(\bar{x}), x-\bar{x}) \geq 0 \quad \text{pour tout } x \in X \tag{9.2.3}$$

Voici l'algorithme que nous proposons:

ALGORITHME 9.2.1

Etape 0. Déterminer un point quelconque $x^1 \in X$.
Poser $k \leftarrow 1$.

Etape 1. Trouver $y^k \in X$ une solution optimale du problème de programmation linéaire:

$$\underset{y\in X}{\text{Max}} \ \underset{i=1,\ldots,k}{\text{Min}} \ (\nabla f(x^i), x^i - y) \qquad (9.2.4)$$

Soit x_o^k la valeur de l'objectif de (9.2.4) à l'optimum.

Etape 2. Critère d'arrêt
Si $x_o^k \leq 0$ alors c'est terminé.

Etape 3. Calcul de x^{k+1}
Calculer $x^{k+1} = x^k + \theta^k (y^k - x^k)$,
où

$$\theta^k = \arg \underset{\theta\in[0,1]}{\text{Min}} \ h(x^k + \theta(y^k - x^k)),$$
$$h(x) = (\nabla f(x), x^k - y^k)^2 .$$

Etape 4. Poser $k \leftarrow k+1$. Retourner à l'étape 1.

Posons

$$v(\theta) = h(x^k + \theta(y^k - x^k)), \quad \theta \in [0,1].$$

Puisque la fonction v est strictement décroissante sur [0,1] en autant que $\nabla f(x)$ est strictement monotone sur X, alors le calcul du pas à l'étape 3 s'avère relativement facile.

Grâce aux résultats du chapitre 2, nous sommes assurés de la convergence de l'algorithme en accord avec le critère d'arrêt choisi. De plus, nous pouvons montrer la proposition suivante:

PROPOSITION 9.2.1

En supposant la stricte monotonie de $\nabla f(x)$ sur $\mathcal{Q}$, si l'algorithme termine après k itérations, x^k est une solution de (9.2.3); sinon, il existe une sous-suite $\{x^k\}$ convergeant vers cette solution. □

Des conditions autres que la monotonie de $\nabla f(x)$ sur X pourraient être envisagées dans la proposition 9.2.1 mais notre intérêt se porte ailleurs.

Dans la prochaine section, nous examinerons brièvement la situation où la fonction f est convexe mais non-nécessairement différentiable partout.

9.3 CAS SPECIAL D'UN PROBLEME DE PROGRAMMATION CONVEXE NON-DIFFERENTIABLE PARTOUT

Notre intérêt se portera maintenant sur un problème général de programmation convexe où la fonction objectif n'est pas différentiable partout. Nous considérons de nouveau le problème

$$\underset{x \in X}{\text{Min}}\ f(x) \tag{9.3.1}$$

où f est continue et convexe mais non-différentiable partout sur X, X étant un polyèdre.

Nous aurions pu imposer des hypothèses plus faibles sur f comme la pseudo-convexité ou la quasi-convexité et examiner de nouveau les résultats de la section 9.2, mais notre objectif ici est plutôt de développer une approche pour résoudre (9.3.1) dans le cas non-différentiable.

Puisque f est continue et convexe, alors une condition nécessaire et suffisante d'optimalité de (9.1.3) est (Rockafellar, 1972, p.213 à 218; Auslender, 1976, p.57):

$$\exists \bar{x} \in X,\ \max_{\bar{c} \in \partial f(\bar{x})} (\bar{c}, x-\bar{x}) \geq 0 \quad \text{pour tout } x \in X \tag{9.3.2}$$

ce qui s'écrit:

$$\underset{x \in X}{\text{Max}}\ \underset{y \in X}{\text{Min}}\ \underset{c \in \partial f(x)}{\text{Max}}\ (c, y-x). \tag{9.3.3}$$

Remarquons que $\partial f(x)$ est un ensemble convexe, compact et non-vide pour tout $x \in X$. De plus, ∂f est une multi-application monotone sur X car f est convexe sur X. Nous pouvons facilement adapter la démonstration de la proposition 1.3 du chapitre 7 de Auslender (1976) concernant des applications monotones pour obtenir le résultat suivant.

PROPOSITION 9.3.1

Lorsque f est continue et convexe sur un ensemble convexe compact X, alors nous avons une équivalence entre (9.3.2) et la relation suivante:

$$\exists \bar{x} \in X, \ \max_{c \in \partial f(x)} (c, x-\bar{x}) \geq 0 \quad \text{pour tout } x \in X \tag{9.3.4}$$

Or, (9.3.4) s'écrit encore:

$$\min_{x \in X} g(x) \tag{9.3.5}$$

où

$$g(x) = \max_{y \in X} \max_{c \in \partial f(y)} (c, x-y) \tag{9.3.6}$$

est convexe sur X.

D'où, la résolution de (9.3.1) se ramène à celle de (993.5)-(9.3.6).

La méthode d'accumulation des contraintes développée à la section précédente (algorithme 9.2.1) peut encore être utilisée pour résoudre le problème (9.3.5)-(9.3.6) à la seule condition que l'on puisse expliciter facilement l'ensemble des points extrêmes de $\partial f(x)$ pour tout $x \in X$. Il va sans dire aussi que le nombre de points extrêmes du sous-différentiel de f à un point x donné doit être petit.

L'algorithme s'énonce comme suit:

ALGORITHME 9.3.1

1. Initialisation

Nous déterminons un flot réalisable $v^1 \in X$ et posons

$$Z^1 = \{(x_o,u) \mid (c_{1j}, v^1-u) \geq x_o, \ c_{1j} \text{ est un point extrême de } \partial f(v^1), \quad j = 1,\ldots,i_1, \ u \in X\},$$

$K \leftarrow 1$.

2. Résolution du sous-problème linéaire

Déterminer une solution optimale (x_o^k, u^k) de

$$\max_{(x_o,u) \in Z^k} x_o$$

3. Critère d'arrêt

Si $x_0^k \leq 0$, c'est terminé.

4. Calcul de l'ensemble Z^{k+1}.

Il s'agit de calculer

$$v^{k+1} = v^k + \theta^k(u^k - v^k),$$

où

$$e(v^{k+1}) = \min_{\theta \in [0,1]} e(v^k + \theta(u^k - v^k)),$$

et

$$e(v) = \{\max_{c \in \partial f(v)} (c, v^k - u^k)\}^2.$$

Puis, nous formons

$$Z^{k+1} = Z^k \cap H(v^{k+1})$$

où

$$H(v^{k+1}) = \{(x_o,u)\} \mid (c_{k+1,j}, v^{k+1}-u) \geq x_0,\ c_{k+1,j} \text{ est un point extrême de } \partial f(v^{k+1}),\ j = 1,\ldots,i_{k+1},\ u \in X\}$$

et retournons à l'étape 2 avec $k \leftarrow k+1$.

Sachant que $h(\theta) = \max_{c \in \partial f(x+\theta(y-x))} (c, x-y)$, $x \neq y$ est continue et décroissante sur $[0,1]$, la fonction e à l'étape 4 est adéquate. De plus, nous pouvons adapter la démarche de la section 9.2 pour démontrer la convergence de cet algorithme vers une solution optimale de (9.3.1).

La fonction f pourrait par exemple prendre la forme suivante:

$$f(x) = \max_{i = 1,\ldots,l} f_i(x)$$

où $f_i(x)$ est supposée continuement différentiable et convexe sur X. La quantité l ne doit pas être trop grande.

Il serait fort intéressant de comparer notre méthode d'accumulation des contraintes avec d'autres algorithmes déjà existants, pour résoudre un problème de programmation convexe général.

CONCLUSION

Ce travail représente une contribution importante au problème de la recherche d'un flot d'équilibre sur un réseau routier. Grâce à des hypothèses de monotonie seulement sur la fonction de délai, nous avons proposé une formulation complémentaire à l'inégalité variationnelle de Smith (1979). La recherche d'un flot d'équilibre se ramenait alors à la résolution d'un problème de programmation convexe ou encore, le calcul d'un point de selle d'une fonction donnée.

Nous avons examiné différentes approches algorithmiques basées sur ces nouvelles formulations. Une nouvelle méthode de coupe pour des problèmes de grandes tailles a été développée et implantée sur ordinateur. Nous avons comparé avantageusement notre méthode avec plusieurs algorithmes pour résoudre le problème de Beckmann.

Sans l'hypothèse de symétrie, le problème devient plus difficile car la résolution exacte de chaque sous-problème linéaire est trop coûteuse. Nous avons développé des variantes à notre algorithme de coupe pour pallier à ces inconvénients.

Des variantes de notre algorithme de base ont été développées grâce à la programmation semi-infinie. De même, différentes méthodes de sous-gradients, de projections, etc. ont été examinées grâce à une formulation primale-duale de notre problème.

Il nous a semblé important aussi d'examiner en détail le comportement de ces formulations d'équilibre en l'absence d'hypothèses de monotonie sur les fonctions de coût. Il nous a été possible de fournir de nouvelles conditions d'existence et d'unicité de l'équilibre différentes de celles de Smith.

Un autre point d'intérêt est le lien qui existe entre notre méthode d'accumulation des contraintes et l'algorithme de Frank & Wolfe. La non-convergence de ce dernier dans le cas asymétrique a été examinée en détail.

Enfin nous avons considéré deux extensions possibles à notre méthode de coupe; la première consiste à résoudre un problème d'affectation où la demande est élastique tandis que la seconde concerne la résolution d'un problème général de programmation convexe en présence d'un ensemble réalisable convexe compact.

APPENDICE A

ESSAI SUR UN RESEAU DE PETITE TAILLE (S.NGUYEN & C.DUPUIS, 1981)

Considérons le réseau illustré à la figure A.1. On utilise pour toutes les expériences la même matrice de demande donnée au tableau A.1.

Dans un premier temps, nous considérons des coûts de transport standard (voir tableau A.2):

$$C_i(x) = C_i(x_i) \text{ pour tout } i.$$

Dans ce cas, on peut obtenir le flot d'équilibre en résolvant le problème convexe de Beckmann et al. Nous contrastons alors la performance de l'algorithme d'accumulation des contraintes avec celle d'un algorithme basé sur la méthode de Frank et Wolfe (1956) (Nguyen, 1976).

Nous examinons ensuite le cas des coûts de transport affinés non-symétriques:

$$C(x) = Ax + b$$

où A est une matrice définie positive mais non-symétrique.

Le tableau A.3 met en évidence la convergence expérimentale de la méthode d'accumulation des contraintes lors de la résolution du problème d'équilibre précédent avec des coûts de transport standard $C_i(x_i)$. La performance de l'algorithme de coupes, dans cet exemple, est nettement supérieure à celle de la méthode de Frank et Wolfe (tableau A.3). Il faut cependant noter que la méthode de Frank-Wolfe n'est pas la meilleure pour des problèmes de petites tailles, elle est surtout destinée aux problèmes réels de grandes tailles où un gain de quelques pourcents de la fonction objectif, en général, n'est pas significatif au niveau des flots sur les arcs. Toutefois, il est encourageant de constater que la méthode d'accumulation des contraintes, conçue principalement pour résoudre des problèmes d'affectation où les coûts sur les arcs sont non-symétriques peut se comparer favorablement aux méthodes élaborées pour des problèmes "symétriques".

Nous passons maintenant aux coûts affinés non-symétriques décrits au tableau A.4.

Le tableau A.5 rapporte les résultats obtenus respectivement avec la méthode d'accumulation des contraintes et l'algorithme de Zuhovickii et al. Le choix systématique de $\theta^k = 1$ dans le calcul du pas pour générer la coupe suivante, est la source de la lenteur de ce dernier par rapport à l'algorithme d'accumulation des contraintes. Il est fort intéressant de constater que le nombre de contraintes actives des sous-problèmes linéaires cesse d'augmenter après quelques itérations. Cette propriété est essentielle si l'on veut que la méthode soit applicable aux problèmes de grandes tailles. Notons aussi que le nombre de pivots nécessaires à la réoptimisation des sous-problèmes linéaires est remarquablement bas. Chaque fois que le nombre de contraintes est inférieur ou égal à celui de l'itération suivante, il y a au moins un pivot impliquant une variable d'écart, et un tel pivot ne nécessite pas la recherche de plus courts chemins. On constate donc sur le tableau A.5, qu'après 16 itérations de l'algorithme d'accumulation des contraintes, on effectue au plus 14 recherches de plus courts chemins (26 pivots au total moins 12 = 14). Dans le troisième chapitre, nous avons vu que cette caractéristique demeure lorsque la taille du problème augmente, ainsi nous avons résolu les problèmes de grandes tailles de manière raisonnablement efficace.

En faisant varier la diagonale de la matrice des coefficients A (tableau A.6), de manière à en faire une matrice définie positive, non-diagonale dominante, on obtient les résultats du tableau A.7, résultats très semblables au cas de la matrice A diagonale dominante.

Nous avons aussi résolu le problème test avec la méthode de diagonalisation. Les résultats sont rapportés au tableau A.8. Le sous-problème "symétrique" interne est d'abord résolu avec la méthode de Frank & Wolfe. Dans ce cas, la méthode complète est remarquablement simple à mettre en programme d'ordinateur. Toutefois, on constate immédiatement que la convergence de la méthode de diagonalisation dépend de manière critique de la précision des solutions que la méthode de Frank-Wolfe peut fournir. Jugeant par la valeur de $B(x^k)$, une borne supérieure de x_o^k, on peut constater sur le tableau A.8 qu'on est encore assez loin de l'équilibre après 2000 itérations de la méthode Frank-Wolfe (20 itérations externes par 100 itérations internes).

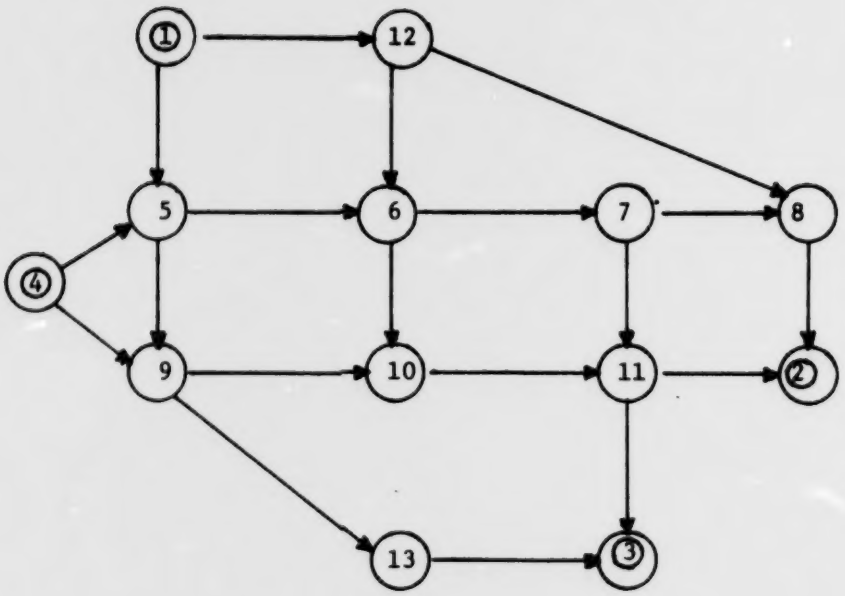


Figure A.1 - Réseau utilisé

$[d_{pq}] =$

p \ q	2	3
1	400	800
4	600	200

Tableau A.1

Matrice de demande

Arc	Origine	Destination	a_{ij}	b_{ij}
1	1	5	0.0125	7
2	1	12	0.01	9
3	4	5	0.01	9
4	4	9	0.005	12
5	5	6	0.0075	3
6	5	9	0.0075	9
7	6	7	0.0125	5
8	6	10	0.005	13
9	7	8	0.0125	5
10	7	11	0.0125	9
11	8	2	0.0125	9
12	9	10	0.005	10
13	9	13	0.005	9
14	10	11	0.0025	6
15	11	2	0.005	9
16	11	3	0.01	8
17	12	6	0.0025	7
18	12	8	0.01	14
19	13	3	0.01	11

$$C_{ij}(x_{ij}) = a_{ij}\, x_{ij} + b_{ij}$$

Tableau A.2

Description des arcs et des coûts "symétriques"

	Algorithme de coupes			Frank & Wolfe		
Nombre d'itérations k	x_0^k	$F(x^k)$	Temps (CPU sec.)	$B(x^k)$	$F(x^k)$	Temps (CPU sec.)
1	100 400,00	94 253,37	0,016	100 400,00	94 253,37	0,014
2	25 285,43	86 847,30	0,041	25 285,43	86 847,30	0,030
3	7 189,15	86 294,06	0,071	7 896,72	86 447,56	0,043
4	2 451,38	86 082,69	0,093	2 548,66	86 229,62	0,058
5	2 284,28	85 963,00	0,120	2 890,46	85 939,96	0,073
10	669,05	85 210,60	0,334	2 454,96	85 555,79	0,146
15	155,94	85 061,23	0,495	1 310,64	85 392,72	0,218
20	27,48	85 035,67	0,640	728,97	85 329,47	0,291
25	3,90	85 030,29	0,750	531,84	85 273,72	0,363
30	0,90	85 028,53	0,862	493,84	85 241,00	0,452
35	0,29	85 028,14	0,975	489,71	85 212,33	0,501
100	---	----	---	162,05	85 102,52	1,446

x^k : solution courante,

$B(x^k) : \max_{y \in \Omega} (C(x^k), x^k - y)$,

$F(x^k) : \sum_i \int_0^{x_i^k} C_i(z)dz$: fonction objectif du problème de Beckmann,

x_0^k : fonction objectif du sous-problème linéaire à l'étape k,

Tableau A.3 - Résultats lorsque les coûts sont "symétriques"

$$
A = \begin{bmatrix}
18. & 10. & 5. & & 1. & 1.2 & & & & & & & & & & & & & \\
1. & 19. & & & & & & & & & & & & & & & & .2 & .1 \\
2. & & 15. & 9. & 2. & 1.1 & & & & & & & & & & & & & \\
 & & 5. & 11. & & 2. & & & & & & .3 & .1 & & & & & & \\
3. & & 2. & & 13.5 & 6. & .4 & 1. & & & & & & & & & & 1. & \\
2. & & 1. & .9 & 2. & 17.5 & & & & & & .4 & .2 & & & & & & \\
 & & & & .6 & & 12.5 & .1 & 4. & 4.4 & & & & & & & & .3 & \\
 & & & & .5 & & 2. & 5.5 & & & & 1.1 & & .1 & & & & .4 & \\
 & & & & & & .5 & & 13.5 & 1. & .6 & & & & & & & & 5. \\
 & & & & & & .2 & & 5. & 33.3 & & & & 2. & 13. & 12. & & & \\
 & & & & & & & & .2 & & 12.5 & & & & 3. & & & & 2. \\
 & & & .5 & & 4. & & 2. & & & & & 9.8 & 2. & .3 & & & & \\
 & & & .1 & & 2. & & & & & & & 1. & 5. & & & & & 1. \\
 & & & & & & & .3 & & 10. & & 3. & & 37. & & .1 & 14. & & \\
 & & & & & & & & & .1 & 11. & & & .2 & 20. & & .5 & & \\
 & & & & & & & & & .2 & & & & .1 & 2. & 50. & & & 6. \\
 & 4. & & & 3. & & .3 & 2. & & & & & & & & & & 20. & 1. \\
 & 4. & & & & & & & 4. & & .3 & & & & & & & 2. & 11. \\
 & & & & & & & & & & & & 2. & & & 5. & & & 10.
\end{bmatrix}
$$

$$C(x) = 10^{-3} Ax \pm b$$

vecteur	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
b^{ϵ} =	7	9	9	12	3	9	5	13	5	9	9	10	9	6	9	8	7	14	11

Tableau A.4 - Description des coûts non-symétriques(diagonale dominante)

	Algorithme de coupes				Zuhovickii et *al.*			
Nombre d'itérations k	x_0^k	Nombre de contraintes	Nombre de pivots	Temps cumulés (CPU sec.)	x_0^k	Nombre de contraintes	Nombre de pivots	Temps cumulés (CPU sec.)
1	176 492,00	1	0	0,026	176 492,00	1	0	0,020
2	22 448,71	2	2	0,046	47 480,00	2	2	0,037
3	12 156,66	2	2	0,070	39 037,35	2	3	0,061
4	3 666,34	2	3	0,101	27 873,95	3	1	0,075
5	2 685,25	3	2	0,127	18 724,85	3	4	0,114
6	1 511,72	4	1	0,146	14 603,53	4	1	0,130
7	972,88	4	1	0,168	5 241,65	4	2	0,146
8	715,99	4	1	0,190	4 303,21	3	2	0,169
9	195,13	4	3	0,222	3 546,59	4	1	0,186
10	133,25	4	3	0,255	1 401,26	4	2	0,204
11	87,26	4	2	0,278	1 377,54	3	2	0,227
12	71,98	4	1	0,300	1 347,94	4	1	0,245
13	30,82	4	1	0,322	1 304,29	4	1	0,260
14	25,58	4	1	0,343	1 137,41	4	1	0,278
15	0,03	4	1	0,364	1 092,53	4	1	0,294
16	0,01	4	2	0,387	1 045,68	4	1	0,312
17					996,35	4	1	0,330
18					943,82	4	1	0,346
19					886,93	4	1	0,361
20					823,66	4	1	0,377
25					241,37	4	1	0,460
30					33,32	4	1	0,544
35					28,18	4	1	0,627

Tableau A.5

Résultats obtenus lorsque les coûts sont non-symétriques, affines et le Jacobien de la fonction de coût est diagonale dominant.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A_{ii} =	12,5	10	10	5	7,5	7,5	12,5	5	12,5	12,5	12,5	5	5	2,5	5	10	2,5	10	10

Tableau A.6

Description des coûts non-symétriques(diagonale non-dominante)

	Algorithme de coupes			
Nombre d'itérations k	x_0^k	Nombre de contraintes	Nombre de pivots	Temps cumulés (CPU sec.)
1	113 420,00	1	0	0,023
2	14 909,32	2	2	0,046
3	1 731,35	2	2	0,068
4	773,63	2	2	0,091
5	373,64	3	1	0,111
6	285,58	3	2	0,138
7	178,94	4	1	0,159
8	113,39	4	3	0,198
9	90,73	4	3	0,238
10	26,94	4	1	0,258
11	8,23	4	1	0,282
12	0,76	4	3	0,313
13	0,70	4	1	0,336
14	0,40	4	2	0,359
15	0,33	4	1	0,380
16	0,03	4	2	0,401
17	0,01	3	2	0,430

Tableau A.7

Résultats obtenus lorsque les coûts sont "non-symétriques" et affines, et le Jacobien de la fonction de coût est non-diagonale dominante.

Nombre d'itérations k	Diagonalisation (FW 50)[1]		Diagonalisation (FW 100)[1]		Diagonalisation (algorithme de coupes)[2]	
	$B(x^k)$	Temps cumulés (CPU sec.)	$B(x^k)$	Temps cumulés (CPU sec.)	$B(x^k)$	Temps cumulés (CPU sec.)
1	36 723,09	0,213	36 198,34	0,434	1 603,81	0,685 (32)
2	5 043,89	0,453	5 420,02	0,879	726,19	0,904 (12)
3	395,57	0,692	272,97	1,325	27,63	0,959 (3)
4	130,89	0,924	167,15	1,772	6,50	1,005 (3)
5	101,46	1,171	76,75	2,226	0,27	1,057 (3)
6	91,79	1,413	67,69	2,677	0,05	1,092 (2)
7	70,09	1,647	56,49	3,121	0,00	1,140 (3)
8	68,06	1,883	48,86	3,560	----	-------
9	54,29	2,123	34,01	4,005		
10	54,07	2,358	38,43	4,444		
11	44,34	2,591	34,73	4,890		
12	44,88	2,830	31,68	5,340		
13	37,49	3,061	29,13	5,789		
14	38,37	3,304	26,96	6,230		
15	32,48	3,544	25,09	6,675		
16	33,52	3,782	23,47	7,115		
17	28,65	4,017	22,04	7,557		
18	29,76	4,262	20,78	8,004		
19	25,64	4,500	19,65	8,438		
20	26,76	4,736	18,64	8,887		

$$B(x^k) = \max_{y \in \Omega} (C(x^k), x^k - y)$$

[1] FW α = la solution x^k avec coûts symétriques (diagonalisés) est obtenue après α itérations de la méthode de Frank et Wolfe.

[2] L'algorithme de coupes est utilisé pour résoudre le sous-problème symétrique; on s'arrête lorsque $x_0^k < 0{,}01$.

Tableau A.8

Résultats obtenus lorsque les coûts sont non-symétriques et affines, et le Jacobien est diagonale dominant.

Afin d'avoir une préciaion quasi exacte des solutions du sous-problème "symétrique", on avait substitué l'algorithme d'accumulation des contraintes à l'algorithme Frank-Wolfe. Dans ce cas, la convergence est très rapide. Le flot d'équilibre est obtenu après sept itérations externes seulement et un nombre total de 58 pivots de l'algorithme d'accumulation des contraintes. Ce dernier nombre est quand même très supérieur à celui pris par l'algorithme d'accumulation des contraintes pour résoudre directement l'exemple.

Il nous semble donc, d'après les essais effectués, que l'algorithme d'accumulation des contraintes représente un bon compromis entre la simplicité de la méthode de diagonalisation couplée avec l'algorithme Frank-Wolfe et l'excellente convergence de la méthode de diagonalisation couplée avec tout autre algorithme qui produit une solution exacte des sous-problèmes convexes.

S. Lawphongpanich et D.W. Hearn ont aussi résolu ces 3 problèmes d'affectation (correspondant respectivement au Jacobien de la fonction de coût symétrique, non-symétrique mais diagonale dominante, non-symétrique et non-diagonale dominante) à l'aide d'une méthode de décomposition simpliciale. Les résultats obtenus semblent supérieurs aux nôtres mais, étant donné que les tests ont été faits sur des machines différentes (temps d'exécution, précision des calculs, solution initiale de départ, etc.), toute comparaison est à éviter.

APPENDICE B

UN EXEMPLE DE STEENBRINK (1974)

Nous avons résolu un exemple de Steenbrink (1974), comportant 9 sommets, 36 arcs et 12 paires O/D et dont les coûts $C_i(x_i)$ sont linéaires:

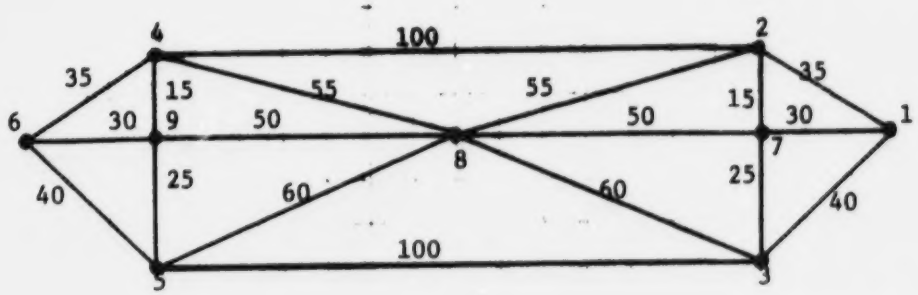


Figure B.1 - Réseau de Steenbrink (1974)

	2	3	4	5
2	0	2000	2000	1000
3	200	0	1000	2000
4	200	100	0	1000
5	100	200	100	0

Tableau B.1 - Matrice O/D de cet exemple

L'exemple a été conçu pour rendre la résolution particulièrement ardue. Nous avons obtenu une valeur de la fonction objectif égale à 16957,809 après 100 itérations de l'algorithme d'accumulation des contraintes durant 42,06 secondes de temps CPU. Steenbrink a obtenu une valeur de 16970 à l'aide d'un programme conçu spécialement pour les problèmes quadratiques. Notre approche s'est donc avérée un outil très valable.

APPENDICE C

Programme Fortran qui permet de déterminer le flot d'équilibre sur le réseau de Hull grâce à la méthode d'accumulation des contraintes

```
      PROGRAM PGGUB(INPUT,OUTPUT,TAPE8,TAPE9)
C
C     ----- NPOD     = NOMBRE DE PAIRES O/D
C     ----- NNODE    = NOMBRE DE SOMMETS.
C     ----- NLINK    = NOMBRE D'ARCS.
C     ----- NOLAB(I) = CONTIENT L'ADRESSE DU PREMIER DES ARCS
C                      (X,Y)  (POUR CHACUN DES X)
C     ----- ND(I)    = L'EXTREMITE TERMINALE DE L'ARC I.
C     ----- MAXCOUP = NOMBRE MAXIMUM DE COUPES RETENUES A CHAQUE
C                     ITERATION
C     ----- VCUM    = FLOT DE L'ITERATION PRECEDENTE
C     ----- KPILEX,POSX,NONPOSX.  GESTION DU FICHIER 8.
C     ----- KPILEY,POSY,NONPOSY.  GESTION DU FICHIER 9.
C
      COMMON /NET/NNODE,NLINK,NOLAB(502),ND(800),NO(800)
      COMMON/NET1/DIST(800),SPDO(800),WIDTH(800),
     +            CAP(800),LTYP(800)
      COMMON /POD/LINIT(150),LTERM(150),DEMANDE(150)
      COMMON /TYL/VCUMT(800)
      COMMON/CHEM/LPR(502),TRAV(800)
      COMMON/INV/BINV(40,40),INDBL(40),IRANG,XOK,VECTB(40)
      COMMON/MULT/RMULT(40)
      COMMON/XI/VCUM(800),FLOW(800)
      COMMON /KPX/KPILEX(20),KARRX(20)
      COMMON /KPY/KPILEY(20),KARRY(20)
      COMMON/COEF/ALPHA(5),BETA(5)
      COMMON/DISK/INDEX1(40),INDEX2(40)
      COMMON WKMEM(1004)
      INTEGER POSX,POSY,PARIMP
      DATA MAXCOUP/20/,EPSILON/-0.001/
      DATA (ALPHA(I),I=1,5)/.73,.61,.76,.68,.68/
      DATA (BETA(I),I=1,5)/3.6,3.5,5.0,5.15,4.50/
C
C-----OUVERTURE DES FICHIERS.
C
      CALL OPENMS(8,INDEX1,40,1)
      CALL OPENMS(9,INDEX2,40,1)
C
C-----CRITERES D'ARRET DE GUB.
C----PARAMETRE "OMEGA" DE LA FONCTION OBJECTIVE.
C---PARAMETRES D'IMPRESSION.
C
      CALL PARGUB(MAXITER,GAPMIN,OMEGA,PARIMP,ISER,ISYM,LPAS)
C
C----INITIALISATION.
C
      NOITE=0
      CALL SECOND(TBEGIN)
C
C---LIRE LES DONNEES DU RESEAU.
C
      CALL RDNET(NPOD)
C
C----LIRE LES DONNEES POUR CHAQUE PAIRE O/D.
C
      CALL RDOD(NPOD)
C
C-----IMPRESSION DES DONNEES DU RESEAU.
C
      CALL SECOND(TTOUT)
      TTOUT=TTOUT-TBEGIN
```

```
      PRINT 150,TTOUT
  150 FORMAT(1HO,//,10X,"TEMPS ECOULE A LIRE LES DONNEES = ",F10.6)
*     IF(PARIMP.EQ.2.OR.PARIMP.EQ.3) THEN
*      CALL IMPNET(PARIMP)
*      CALL IMPOD(NPOD)
*     ENDIF
      CALL SECOND(TBEGIN)
*
*-----INITIALISATION DES POINTEURS
*
      POSX=POSY=0
      NONPOSX=NONPOSY=1
      DO 300 I=1,MAXCOUP
  300 KPILEX(I)=KPILEY(I)=I+1
      KPILEX(MAXCOUP)=KPILEY(MAXCOUP)=0
*
*-----DETERMINATION D'UN FLOT INITIAL DE DEPART
*
      DO 400 J=1,NLINK
  400 VCUMT(J)=VCUM(J)=0.0
      DO 500 J=1,NLINK
  500 TRAV(J)=UCOST(J)
      CALL LOAD(NPOD,FY)
*
*-----EMMAGASINER LA SOLUTION REALISABLE INITIALE ET L'IMPRIMER
*     S'IL Y A LIEU.
*
      DO 600 I=1,NLINK
  600 VCUM(I)=VCUMT(I)
      CX1X1=0.0
      DO 650 I=1,NLINK
      TRAV(I)=UCOST(I)
  650 CX1X1=CX1X1+TRAV(I)*VCUM(I)
      CALL AJUSTX(POSX,NONPOSX)
      CALL WRITMS(8,TRAV,NLINK,POSX)
      CALL SECOND(TTOUT)
      TIME=TTOUT-TBEGIN
*
      IF(PARIMP.NE.1.AND.PARIMP.NE.3) GO TO 660
      CALL IMPRESU(NOITE,NLINK,XOK,XOK,THETA,TIME,1,VN,LPITER,0)
      IF(ISYM.EQ.1)CALL FOBJ(NLINK)
*
*
*-----ITERATION NO 1: PROBLEME DE PLUS COURT CHEMIN
*
  660 CALL SECOND(TBEGIN)
      NOITE=1
      DO 700 I=1,NLINK
  700 VCUMT(I)=0.0
      CALL LOAD(NPOD,FY)
*
*-----CALCUL DU PAS
*
      DO 750 I=1,NLINK
  750 FLOW(I)=VCUMT(I)-VCUM(I)
      CALL NEWTON(THETA,NLINK)
*
      CX2X2=CX2Y1=0.0
      DO 900 I=1,NLINK
      TRAV(I)=UCOST(I)
      CX2X2=CX2X2+TRAV(I)*VCUM(I)
```

```
      900  CX2Y1=CX2Y1+TRAV(I)*VCUMT(I)
           CALL AJUSTX(POSX,NONPOSX)
           CALL WRITMS(8,TRAV,NLINK,POSX)
           CALL AJUSTY(POSY,NONPOSY)
           CALL WRITMS(9,VCUMT,NLINK,POSY)
C
           CALL SECOND(TTOUT)
           TIME=TIME+TTOUT-TBEGIN
           VN=0.0
           DO 930 I=1,NLINK
      930  VN=VN+(FLOW(I)-VCUMT(I)+VCUM(I))**2
           XOK=CX1X1-FY
           DO 940 I=1,NLINK
      940  VCUMT(I)=0.0
           CALL LOAD(NPOD,PCC)
           PCC=CX2X2-PCC
           IF(PARIMP.NE.1.AND.PARIMP.NE.3) GO TO 945
           CALL IMPRESU(NOITE,NLINK,XOK,PCC,THETA,TIME,2,VN,0,0)
           IF(ISYM.EQ.1)CALL FOBJ(NLINK)
C
      945  CALL SECOND(TBEGIN)
           NOITE=2
C
C-----ITERATIONS SUBSEQUENTES
C     - INITIALISATION DE L'ALGORITHME GUB.
C
           CALL INIT(FY,CX2Y1,CX2X2)
           LPEC=LPITER=0
           IPASSE=0
C
C------CALCUL DES MULTIPLICATEURS
C
      950  IF(INDBL(IRANG).NE.-999999) GO TO 1002
           DO 1000 I=1,IRANG
      1000 RMULT(I)=BINV(IRANG,I)-OMEGA*BINV(1,I)
           GO TO 1006
      1002 OMEG=OMEGA
           IF(OMEG.EQ.0.0) OMEG=1.0
           DO 1004 I=1,IRANG
      1004 RMULT(I)=-BINV(1,I)*OMEG
C
C
C-----COLONNE ENTRANTE.
C
      1006 VMAX=-1.0E10
           IR=IRANG-1
           DO 1010 I=2,IR
           IF(RMULT(I).LE.VMAX)GO TO 1010
            VMAX=RMULT(I)
            JS=I
C
      1010 CONTINUE
C
C----S'IL EXISTE UN MULTIPLICATEUR STRICTEMENT POSITIF, UNE
C     VARIABLE D'ECART ENTRE DANS LA BASE.
C
           IF(VMAX.LE.0.0)GO TO 1017
            IS=IRANG+2
            DO 1015 I=1,IRANG
      1015  BINV(I,IS)=BINV(I,JS)
            LPEC=LPEC+1
```

```
                      IECH=0
185                   GO TO 1075
                *
                *
                *-----CALCUL D'UN FLOT TOUT-OU-RIEN
                *
190              1017 IECH=1
                      DO 1020 I=1,NLINK
                 1020 TRAV(I)=VCUMT(I)=0.0
                      IPTR=POSX
                      INR=IRANG
195              1025 CALL READMS(8,FLOW,NLINK,IPTR)
                      DO 1030 I=1,NLINK
                 1030 TRAV(I)=TRAV(I)-RMULT(INR)*FLOW(I)
                      IPTR=KPILEX(IPTR)
                      INR=INR-1
200                   IF (INR.GT.1)GOTO 1025
                      CALL LOAD(NPOD,FY)
                      IF((FY-RMULT(1)).GE.EPSILON)GOTO 3000
                *
                *-----CRITERE DE SORTIE
205             *
                      BINV(1,IRANG+1)=1.0
                      IPTR=POSX
                      INR=IRANG
                 1040 SOM=0.0
210                   CALL READMS(8,FLOW,NLINK,IPTR)
                      DO 1050 I=1,NLINK
                 1050 SOM=SOM+FLOW(I)*VCUMT(I)
                      BINV(INR,IRANG+1)=SOM
                      IPTR=KPILEX(IPTR)
215                   INR=INR-1
                      IF(INR.GT.1)GO TO 1040
                      IS=IRANG+2
                      DO 1070 I=1,IRANG
                      SOM=0.0
220                   DO 1060 J=1,IRANG
                 1060 SOM=SOM+BINV(I,J)*BINV(J,IRANG+1)
                 1070 BINV(I,IS)=SOM
                 1075 VMIN=1.0E10
                      DO 1080 I=2,IRANG
225                   IF(BINV(I,IS).LE.0.0) GO TO 1080
                      PIVOT=VECTB(I)/BINV(I,IS)
                      IF(PIVOT.GT.VMIN)GO TO 1080
                       VMIN=PIVOT
                       JR=I
230             *
                 1080 CONTINUE
                *
                *-----REINITIALISATION DES INDICES DE BASE.
                *
235                   IF(IECH.NE.1) GO TO 1081
                       CALL AJUSTY(POSY,NONPOSY)
                       CALL WRITMS(9,VCUMT,NLINK,POSY)
                *
                 1081 IF(INDBL(JR) .LE.0) GO TO 1084
240                    KK=NONPOSY
                       NONPOSY=INDBL(JR)
                       IF(KARRY(NONPOSY).EQ.0)GO TO 1082
                       KPILEY(KARRY(NONPOSY))=KPILEY(NONPOSY)
                      GO TO 1083
```

```
      1052 POSY=KPILEY(NONPOSY)
*
      1083 IF(KPILEY(NONPOSY).NE.0)
     1    KARRY(KPILEY(NONPOSY))=KARRY(NONPOSY)
           KPILEY(NONPOSY)=KK
*
      1084 INDBL(JR)=POSY
           IF(IECH.EQ.0)INDBL(JR)=-JS
*
*
<-- --ON EFFECTUE LE PIVOT.
*
           PIVOT=BINV(JR,IS)
           DO 1090 I=1,IRANG
      1090 BINV(JR,I)=BINV(JR,I)/PIVOT
           VECTB(JR)=VECTB(JR)/PIVOT
           DO 2020 I=1,IRANG
           IF(I.EQ.JR) GO TO 2020
           PIVOT=BINV(I,IS)
           IF(PIVOT.EQ.0.0)GO TO 2020
           DO 2010 J=1,IRANG
      2010 BINV(I,J)=BINV(I,J)-PIVOT*BINV(JR,J)
           VECTB(I)=VECTB(I)-PIVOT*VECTB(JR)
      2020 CONTINUE
           LPITER=LPITER+1
           IF(INDBL(IRANG).EQ.-999999)GO TO 950
           IPASSE=IPASSE+1
           IF(IPASSE.NE.2)GO TO 950
*
<-----DETERMINATION DE LA SOLUTION OPTIMALE DU PROB. LINEAIRE.
*
      3000 OMEG=OMEGA
           IPASSE=0
           CALL OPT(NLINK,OMEGA)
*          CALL OPT1(NPOD,OMEGA,POSX)
           IF(OMEG.EQ.OMEGA) GO TO 3600
           PRINT 3500,OMEGA
      3500 FORMAT(1H ,10X,"VALEUR DE OMEGA= ",F16.5)
           GO TO 950
*
*
<-----CALCUL DU PAS.
*
      3600 DO 3700 I=1,NLINK
      3700 FLOW(I)=VCUMT(I)-VCUM(I)
           CALL NEWTON(THETA,NLINK)
*
           CXIXI=0.0
           DO 5000 I=1,NLINK
           TRAV(I)=UCOST(I)
      5000 CXIXI=CXIXI+TRAV(I)*VCUM(I)
           IPIR=POSX
           CALL AJUSTX(POSX,NONPOSX)
           CALL WRITMS(8,TRAV,NLINK,POSX)
           CALL SECOND(TTOUT)
           TIME=TIME+TTOUT-TBEGIN
           VN=0.0
           DO 6000 I=1,NLINK
      6000 VN=VN+(FLOW(I)-VCUMT(I)+VCUM(I))**2
           DO 6100 I=1,NLINK
      6100 VCUMT(I)=0.0
```

```
                      CALL LOAD(NPOD,FY)
                      FY=CXIXI-FY
                      IF(PARIMP.NE.1.AND.PARIMP.NE.3) GO TO 6200
                      CALL IMPRESU(NOITE,NLINK,XOK,FY,THETA,TIME,IRANG,VN,LPITER,LPEC)
  310                 IF(ISYM.EQ.1) CALL FOBJ(NLINK)
                    *
                 6200 IF(XOK.LE.GAPMIN.OR.NOITE.GT.MAXITER)STOP
                      LPEC=LPITER=0
                      CALL SECOND(TBEGIN)
  315                 NOITE=NOITE+1
                    *
                    *-----PURGE LES CONTRAINTES NON-ACTIVES.
                    *
                      IF(ISER.EQ.1)
  320                1CALL PURGE(ISER,POSX,NONPOSX,IPTR)
                    *
                    *-----CALCUL DU NOUVEL INVERSE.
                    *
                      IRANG=IRANG+1
  325                 CALL GENERK(NLINK,CXIXI)
                      CALL NEWINV(NLINK)
                    *
                    *-----REVISER LES INDICES DE BASE: INDICE DE Z
                    *
  330                 INDBL(IRANG)=-999999
                      GO TO 950
                      END
```


SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
10272 PGGUB

VARIABLES	SN	TYPE		RELOCATION					
0	ALPHA	REAL	ARRAY	COEF	5	BETA	REAL	ARRAY	COEF
0	BINV	REAL	ARRAY	INV	4540	CAP	REAL	ARRAY	NET1
11501	CXIXI	REAL			11452	CX1X1	REAL		
11457	CX2X2	REAL			11460	CX2Y1	REAL		
454	DEMANDE	REAL	ARRAY	POD	0	DIST	REAL	ARRAY	NET1
11370	EPSILON	REAL			1440	FLOW	REAL	ARRAY	XI
11451	FY	REAL			11434	GAPMIN	REAL		
11447	I	INTEGER			11471	IECH	INTEGER		
3100	INDBL	INTEGER	ARRAY	INV	0	INDEX1	INTEGER	ARRAY	DISK
50	INDEX2	INTEGER	ARRAY	DISK	11473	INR	INTEGER		
11463	IPASSE	INTEGER			11472	IPTR	INTEGER		
11466	IR	INTEGER			3150	IRANG	INTEGER		INV
11470	IS	INTEGER			11436	ISER	INTEGER		
11437	ISYM	INTEGER			11450	J	INTEGER		
11477	JR	INTEGER			11467	JS	INTEGER		
24	KARRX	INTEGER	ARRAY	KPX	24	KARRY	INTEGER	ARRAY	KPY
11500	KK	INTEGER			0	KPILEX	INTEGER	ARRAY	KPX
0	KPILEY	INTEGER	ARRAY	KPY	0	LINIT	INTEGER	ARRAY	POD
11440	LPAS	INTEGER			11462	LPEC	INTEGER		
11456	LPITER	INTEGER			0	LPR	INTEGER	ARRAY	CHEM
226	LTERM	INTEGER	ARRAY	POD	6200	LTYP	INTEGER	ARRAY	NET1
11367	MAXCOUP	INTEGER			11433	MAXITER	INTEGER		
770	ND	INTEGER	ARRAY	NET	1	NLINK	INTEGER		NET

```
SUBROUTINE AJUSTX      73/1     OPT=0, ROUND= A/ S/ M/-D, -DS      FTN 5.1+564          83/C
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=5C
FTN5.

  1                     SUBROUTINE AJUSTX(POSX, NONPOSX)
  2                     COMMON/KPX/KPILEX(20), KARRX(20)
  3                     INTEGER POSX
  4               *
  5                     IF(POSX. NE. 0)KARRX(POSX)=NONPOSX
  6                     K=KPILEX(NONPOSX)
  7                     KPILEX(NONPOSX)=POSX
  8                     POSX=NONPOSX
  9                     KARRX(POSX)=0
 10                     NONPOSX=K
 11                     RETURN
 12                     END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

  K             32B                          INTEGER
  KARRX         24B   /KPX/                  INTEGER        20
  KPILEX         0B   /KPX/                  INTEGER        20
  NONPOSX        2    DUMMY-ARG              INTEGER
  POSX           1    DUMMY-ARG              INTEGER

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  AJUSTX        5B       2

--STATISTICS--

  PROGRAM-UNIT LENGTH              35B =  29
  CM LABELLED COMMON LENGTH        50B =  40
  CM STORAGE USED               61300B =  25280
  COMPILE TIME                  0.071 SECONDS
```

```
[redacted] SUBROUTINE AJUSTY [redacted] 73/1 [redacted] OPT=0, ROUND= [redacted] A/ S/ M/ D, -DS [redacted] FTN 5.1+564 [redacted] 83/0
DS=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50(
FTN5.

                  SUBROUTINE AJUSTY(POSY,NONPOSY)
                  COMMON/KPY/KPILEY(20),KARRY(20)
                  INTEGER POSY
          *
                  IF(POSY.NE.0)KARRY(POSY)=NONPOSY
                  K=KPILEY(NONPOSY)
                  KPILEY(NONPOSY)=POSY
                  POSY=NONPOSY
                  KARRY(POSY)=0
                  NONPOSY=K
                  RETURN
                  END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

  K          32B                                INTEGER
  KARRY      24B   /KPY/                        INTEGER         20
  KPILEY      0B   /KPY/                        INTEGER         20
  NONPOSY     2    DUMMY-ARG                    INTEGER
  POSY        1    DUMMY-ARG                    INTEGER

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  AJUSTY      5B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH                35B =   29
  CM LABELLED COMMON LENGTH          50B =   40
  CM STORAGE USED                61300B =  25280
  COMPILE TIME                   0.076 SECONDS
```

```
SUBROUTINE LOAD        73/1      OPT=0, ROUND= A/ S/ M/-D, -DS      FTN 5.1+564        83/(
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50
FTN5.

            SUBROUTINE LOAD(NPOD, FY)
      *
      *-----RESOUT UN PROBLEME DE PLUS COURT CHEMIN OU LE COUT CX
      *     SUR LES ARCS EST STOCKE DANS LE VECTEUR TRAV.
      *     EMMAGASINER LE FLOT Y TOUT-OU-RIEN DANS VCUMT.
      *     FY CONTIENDRA LA VALEUR (CX, Y).
      *
      *
            COMMON/NET/NNODE, NLINK, NOLAB(502), ND(800), NO(800)
            COMMON/POD/LINIT(150), LTERM(150), DEMANDE(150)
            COMMON/TYL/VCUMT(800)
            COMMON/CHEM/LPR(502), TRAV(800)
            COMMON WKMEM(1004)
            DIMENSION TSUM(502)
            EQUIVALENCE (TSUM(1), WKMEM(1))
            FY=0.0
            NF=NN=0
            DO 600 I=1, NPOD
              IF(NN.EQ.LINIT(I))THEN
                IF(NF.EQ.LTERM(I))GO TO 580
                CALL PARCOU(NN, NF, GK)
                GO TO 570
            ENDIF
      *
            IF(NN.NE.0)CALL PARCOU(NN, NF, GK)
            NN=LINIT(I)
            CALL MINTREE(NN, 23)
        570 NF=LTERM(I)
            GK=0.0
        580 GK=GK+DEMANDE(I)
            FY=FY+TSUM(NF)*DEMANDE(I)
        600 CONTINUE
            CALL PARCOU(NN, NF, GK)
            RETURN
            END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE      -NAME---ADDRESS--BLOCK-----

  DEMANDE    454B   /POD/                     REAL          150      NN         104B
  FY            2   DUMMY-ARG                 REAL                   NNODE        0B   /NET/
  GK         110B                             REAL                   NO        2430B   /NET/
  I          106B                             INTEGER                NOLAB        2B   /NET/
  LINIT        0B   /POD/                     INTEGER       150      NPOD         -1   DUMMY-ARG
  LPR          0B   /CHEM/                    INTEGER       502      TRAV       766B   /CHEM/
  LTERM      226B   /POD/                     INTEGER       150      TSUM         0B   //
  ND         770B   /NET/                     INTEGER       800      VCUMT        0B   /TYL/
  NF         105B                             INTEGER                WKMEM        0B   //
  NLINK        1B   /NET/                     INTEGER
```

```
      SUBROUTINE INIT(CXIYI,CX2Y1,CX2X2)
*
*-----INITIALISATION DE L'INVERSE DE LA BASE, DES INDICES
*     DES COLONNES DE BASE ET DES INDICES DES LIGNES DE BASE, DU
*     RANG DE LA BASE, DE LA FONCTION OBJECTIVE ET DE LA VALEUR
*     DES VARIABLES DE BASE.
*
      COMMON/INV/BINV(40,40),INDBL(40),IRANG,XOK,VECTB(40)
      IRANG=3
      BINV(1,2)=BINV(2,1)=BINV(3,2)=1.0
      BINV(1,3)=BINV(2,2)=BINV(2,3)=0.0
      BINV(3,3)=-1.0
      BINV(1,1)=-CXIYI
      BINV(3,1)=CX2Y1-CXIYI
*
      VECTB(1)=XOK
      VECTB(2)=1.0
      VECTB(3)=XOK+CX2Y1-CX2X2
*
      INDBL(1)=0
      INDBL(2)=1
      INDBL(3)=-999999
      RETURN
      END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS--	BLOCK	PROPERTIES	TYPE	SIZE
BINV	0B	/INV/		REAL	1600
CXIYI	1	DUMMY-ARG		REAL	
CX2X2	3	DUMMY-ARG		REAL	
CX2Y1	2	DUMMY-ARG		REAL	

-NAME-	--ADDRESS--	BLOCK
INDBL	3100B	/INV/
IRANG	3150B	/INV/
VECTB	3152B	/INV/
XOK	3151B	/INV/

--ENTRY POINTS--(LO=A)

-NAME-	--ADDRESS--	ARGS
INIT	5B	3

--STATISTICS--

PROGRAM-UNIT LENGTH	56B = 46
CM LABELLED COMMON LENGTH	3222B = 1682
CM STORAGE USED	61300B = 25280
COMPILE TIME	0.128 SECONDS

```
SUBROUTINE PURGE      73/1     OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+564          83/0.
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,PL=50
FTN5.

      1                 SUBROUTINE PURGE(ISER,POSX,NONPOSX,MARCH)
      2           *
      3           *-----PURGE LES CONTRAINTES NON-ACTIVES.
      4           *
      5                 COMMON/INV/BINV(40,40),INDBL(40),IRANG,XOK,VECTB(40)
      6                 COMMON/KPX/KPILEX(20),KARRX(20)
      7                 COMMON WKMEM(1004)
      8                 DIMENSION INDBC(40)
      9                 EQUIVALENCE (INDBC(1),WKMEM(1))
     10                 INTEGER POSX
     11           *
     12           *-----ENLEVONS LES LIGNES.
     13           *
     14                 IBAR=2
     15                 DO 20 I=2,IRANG
     16            20   INDBC(I)=0
     17                 DO 100 I=2,IRANG
     18                 IF(INDBL(I).LT.0)THEN
     19                  INDBC(-INDBL(I))=1
     20                  GO TO 100
     21                 ENDIF
     22           *
     23                 IF(IBAR.NE.I) THEN
     24                  DO 50 J=1,IRANG
     25            50    BINV(IBAR,J)=BINV(I,J)
     26                 VECTB(IBAR)=VECTB(I)
     27                 INDBL(IBAR)=INDBL(I)
     28                 ENDIF
     29           *
     30                 IBAR=IBAR+1
     31            100  CONTINUE
     32           *
     33           *-----ENLEVONS LES COLONNES.
     34           *
     35                 IBAR=IBAR-1
     36                 IBAR1=2
     37                 DO 200 I=2,IRANG
     38                 IF(INDBC(I).NE.0)GO TO 200
     39                 IF(IBAR1.NE.I) THEN
     40                  DO 150 J=1,IBAR
     41            150   BINV(J,IBAR1)=BINV(J,I)
     42                 ENDIF
     43           *
     44                 IBAR1=IBAR1+1
     45            200  CONTINUE
     46           *
     47           *-----REINITIALISONS LA PILE PILEX.
     48           *
     49                 DO 300 I=2,IRANG
     50                 J=IRANG-I+2
     51                 IF(INDBC(J).NE.0) THEN
     52                  KK=NONPOSX
     53                  NONPOSX=MARCH
     54                  IF(KARRX(NONPOSX).NE.0)THEN
     55                   KPILEX(KARRX(NONPOSX))=KPILEX(NONPOSX)
     56                  ELSE
     57                   POSX=KPILEX(NONPOSX)
     58                  ENDIF
     59
```

```
      IF (KPILEX(NONPOSX).NE.0)
    1 KARRX(KPILEX(NONPOSX))=KARRX(NONPOSX)
      MARCH=KPILEX(NONPOSX)
       KPILEX(NONPOSX)=KK
       GO TO 300
      ENDIF
*
      MARCH=KPILEX(MARCH)
  300 CONTINUE
      IRANG=IBAR
      RETURN
      END
```

--VARIABLE MAP--(LO=A)

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
BINV	0B	/INV/		REAL	1600
I	234B			INTEGER	
IBAR	233B			INTEGER	
IBAR1	241B			INTEGER	
INDBC	0B	//	EQV	INTEGER	40
INDBL	3100B	/INV/		INTEGER	40
IRANG	3150B	/INV/		INTEGER	
ISER	1	DUMMY-ARG	UNUSED/*S*	INTEGER	
J	237B			INTEGER	

-NAME	---ADDRESS	--BLOCK	-----PR
KARRX	24B	/KPX/	
KK	245B		
KPILEX	0B	/KPX/	
MARCH	4	DUMMY-ARG	
NONPOSX	3	DUMMY-ARG	
POSX	2	DUMMY-ARG	
VECTB	3152B	/INV/	
WKMEM	0B	//	EQ
XOK	3151B	/INV/	

--STATEMENT LABELS--(LO=A)

-LABEL	-ADDRESS	-----PROPERTIES	----DEF
20	INACTIVE	DO-TERM	16
50	INACTIVE	DO-TERM	25
100	73B	DO-TERM	31
150	INACTIVE	DO-TERM	41
200	137B	DO-TERM	45
300	217B	DO-TERM	68

--ENTRY POINTS--(LO=A)

-NAME	---ADDRESS	--ARGS---
PURGE	5B	4

--STATISTICS--

PROGRAM-UNIT LENGTH	250B =	168
CM BLANK COMMON LENGTH	1754B =	1004
CM LABELLED COMMON LENGTH	3272B =	1722
CM STORAGE USED	61300B =	25280
COMPILE TIME	0.367 SECONDS	

```
SUBROUTINE NEWINV      73/1      OPT=0, ROUND= A/ S/ M/-D, -DS      FTN 5.1+564          83/0
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50
FTN5.

                 SUBROUTINE NEWINV(NLINK)
          *
          *-----CALCUL DU NOUVEL INVERSE.
          *
                 COMMON/INV/BINV(40,40), INDBL(40), IRANG, XOK, VECTB(40)
                 COMMON WKMEM(1004)
                 DIMENSION SAUVE(40)
                 EQUIVALENCE (SAUVE(1), WKMEM(1))
                 DO 10 I=1, IRANG-1
                 SAUVE(I)=0.0
                 DO 10 J=1, IRANG-1
           10    SAUVE(I)=SAUVE(I)+BINV(IRANG, J)*BINV(J, I)
                 VECTBK=0.0
                 DO 20 I=1, IRANG-1
                 VECTBK=VECTBK+BINV(IRANG, I)*VECTB(I)
           20    BINV(IRANG, I)=SAUVE(I)
                 VECTB(IRANG)=VECTBK-VECTB(IRANG)
                 RETURN
                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE      -NAME---ADDRESS--BLOCK-----

  BINV        0B   /INV/                       REAL          1600      SAUVE       0B   //
  I         104B                               INTEGER                 VECTB    3152B   /INV/
  INDBL    3100B   /INV/                       INTEGER         40      VECTBK    110B
  IRANG    3150B   /INV/                       INTEGER                 WKMEM       0B   //
  J         106B                               INTEGER                 XOK      3151B   /INV/
  NLINK       1    DUMMY-ARG UNUSED/*S*        INTEGER

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE    DO-TERM         12
    20  INACTIVE    DO-TERM         16

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  NEWINV      5B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH                114B =   76
  CM BLANK COMMON LENGTH            1754B =   1004
  CM LABELLED COMMON LENGTH         3222B =   1682
  CM STORAGE USED                  61300B =   25280
  COMPILE TIME                     0.171 SECONDS
```

[redacted] DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/[redacted]SB/[redacted]/ [redacted]-ID/[redacted]PMD/[redacted]ST,PL=[redacted]000C[redacted]
FT[redacted]

```
      SUBROUTINE OPT(NLINK,OMEGA)
*
*-----EVALUATION DE LA SOL. OPT. DU PROB. LINEAIRE ET
*     DE LA VALEUR DE LA FONCTION OBJECTIVE.
*
      COMMON/XI/VCUM(800),FLOW(800)
      COMMON /TYL/VCUMT(800)
      COMMON/INV/BINV(40,40),INDBL(40),IRANG,XOK,VECTB(40)
      DO 10 I=1,NLINK
   10 VCUMT(I)=0.0
      DO 20 I=1,IRANG
      IF(INDBL(I).EQ.-999999) THEN
       OMEGA=OMEGA/2.0
       RETURN
      ENDIF
*
      IF(VECTB(I).EQ.0.0)GO TO 20
      IF(INDBL(I))20,14,16
   14 XOK=VECTB(I)
      GO TO 20
   16 IN=INDBL(I)
      CALL READMS(9,FLOW,NLINK,IN)
      DO 18 J=1,NLINK
   18 VCUMT(J)=VCUMT(J)+VECTB(I)*FLOW(J)
   20 CONTINUE
      RETURN
      END
```

--VARIABLE MAP--(LO=A)

-NAME-	-ADDRESS-	-BLOCK-	-PROPERTIES-	-TYPE-	-SIZE-
BINV	0B	/INV/		REAL	1600
FLOW	1440B	/XI/		REAL	800
I	121B			INTEGER	
IN	124B			INTEGER	
INDBL	3100B	/INV/		INTEGER	40
IRANG	3150B	/INV/		INTEGER	
J	125B			INTEGER	

-NAME-	-ADDRESS-	-BLOCK-	-PR
NLINK	1	DUMMY-ARG	
OMEGA	2	DUMMY-ARG	
VCUM	0B	/XI/	
VCUMT	0B	/TYL/	
VECTB	3152B	/INV/	
XOK	3151B	/INV/	

--PROCEDURES--(LO=A)

-NAME-	-TYPE-	-ARGS-	-CLASS-
READMS		4	SUBROUTINE

--STATEMENT LABELS--(LO=A)

-LABEL-	-ADDRESS-	-PROPERTIES-	-DEF
10	INACTIVE	DO-TERM	10
14	INACTIVE		19
16	51B		21
18	INACTIVE	DO-TERM	24
20	76B	DO-TERM	25

```
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=5000
FTN5.

 1                SUBROUTINE GENERK(NLINK, CXIXI)
 2           *
 3           *-----GENERER UNE NOUVELLE LIGNE DANS LA BASE
 4           *
 5                COMMON/INV/BINV(40,40), INDBL(40), IRANG, XOK, VECTB(40)
 6                COMMON/CHEM/LPR(302), TRAV(800)
 7                COMMON/XI/VCUM(800), FLOW(800)
 8                COMMON/TYL/VCUMT(800)
 9                BINV(1, IRANG)=0.0
10                BINV(IRANG, 1)=1.0
11                BINV(IRANG, IRANG)=-1.0
12                DO 10 I=2, IRANG-1
13            10  BINV(IRANG, I)=BINV(I, IRANG)=0.0
14                DO 120 I=1, IRANG-1
15                IF(INDBL(I).LE.0)GO TO 120
16                INN=INDBL(I)
17                CALL READMS(9, VCUMT, NLINK, INN)
18                DO 100 J=1, NLINK
19            100 BINV(IRANG, I)=BINV(IRANG, I)+TRAV(J)*VCUMT(J)
20            120 CONTINUE
21                VECTB(IRANG)=CXIXI
22                RETURN
23                END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

  BINV          0B  /INV/                    REAL          1600
  CXIXI          2  DUMMY-ARG                REAL
  FLOW        1440B  /XI/                     REAL           800
  I            123B                          INTEGER
  INDBL       3100B  /INV/                    INTEGER         40
  INN          126B                          INTEGER
  IRANG       3150B  /INV/                    INTEGER
  J            127B                          INTEGER

 -NAME---ADDRESS--BLOCK-----PR
  LPR           0B  /CHEM/
  NLINK          1  DUMMY-ARG
  TRAV        766B  /CHEM/
  VCUM          0B  /XI/
  VCUMT         0B  /TYL/
  VECTB      3152B  /INV/
  XOK        3151B  /INV/

--PROCEDURES--(LO=A)
 -NAME------TYPE--------ARGS------CLASS-----

  READMS                    4     SUBROUTINE

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE    DO-TERM          13
   100  INACTIVE    DO-TERM          19
   120     75B      DO-TERM          20
```

```
[redacted]BRC[redacted]NE [redacted]RCO[redacted]  [redacted]1 [redacted]OPT[redacted]ROU[redacted]= A,[redacted]B/ N,[redacted]5,-DS  [redacted]FTN 5.1+564[redacted]  83/0[redacted]
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50(
FTN5.

      1                 SUBROUTINE PARCOU(NN, NF, GK)
      2           *
      3           *-----PARCOURS D'UN CHEMIN DE L'ORIGINE NN A LA DESTINATION NF
      4           *     A L'AIDE DES VECTEURS LPR ET NO.  STOCKAGE DU FLOT TOUT-
      5           *     OU-RIEN DANS VCUMT.
      6           *
      7                 COMMON/NET/NNODE, NLINK, NOLAB(502), ND(800), NO(800)
      8                 COMMON/TYL/VCUMT(800)
      9                 COMMON/CHEM/LPR(502), TRAV(800)
     10           *
     11                 NFF=NF
     12            80   L=LPR(NFF)
     13                 VCUMT(L)=VCUMT(L)+GK
     14                 NFF=NO(L)
     15                 IF(NFF.NE.NN)GO TO 80
     16                 RETURN
     17                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE      -NAME---ADDRESS--BLOCK-----P

  GK            3   DUMMY-ARG                REAL                    NN            1   DUMMY-ARG
  L            32B                           INTEGER                 NNODE        0B   /NET/
  LPR           0B   /CHEM/                   INTEGER          502    NO        2430B   /NET/
  ND          770B   /NET/                    INTEGER          800    NOLAB        2B   /NET/
  NF            2   DUMMY-ARG                INTEGER                 TRAV       766B   /CHEM/
  NFF          31B                           INTEGER                 VCUMT         0B   /TYL/
  NLINK         1B   /NET/                    INTEGER

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

     80      11B                        12

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  PARCOU       5B      3

--STATISTICS--

  PROGRAM-UNIT LENGTH              35B =   29
  CM LABELLED COMMON LENGTH     10156B =  4206
  CM STORAGE USED               61300B = 25280
  COMPILE TIME                  0.098 SECONDS
```

```
UO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=5000
FTN5.

 1                SUBROUTINE IMPRESU(NOITE, NLINK, XOK, FY, THETA, TIME, IR, VN, LP, LPEC)
 2           *
 3           *-----IMPRESSION DU FLOT STOCKE DANS VCUM A L'ITERATION NOITE.
 4           *
 5                COMMON/XI/VCUM(800), FLOW(800)
 6           *    DATA NPRINT/20/
 7                PRINT 10, NOITE
 8             10 FORMAT('0', 10X, 'ITER. NO. : ', I5, //)
 9           *    IF(NOITE/NPRINT*NPRINT.EQ.NOITE) PRINT 12, (VCUM(I), I=1, NLINK)
10           *12  FORMAT(' ', 10F13.6)
11                PRINT 15, TIME
12             15 FORMAT('0', 10X, 'TEMPS CUMULE:   ', F16.5)
13                IF(NOITE.EQ.0) RETURN
14                PRINT 20, XOK, FY, THETA, IR-1, VN, LP, LPEC
15             20 FORMAT('0', 10X, 'GAP COURANT(PL):    ', F16.5,
16               6/, ' ', 10X, 'GAP(P.C.C.):  ', F16.5,
17               1/, ' ', 10X, 'VALEUR DU PAS:  ', F16.5,
18               2/, ' ', 10X, 'NB. DE COUPES:  ', I5,
19               3/, ' ', 10X, 'NORME:  ', F16.5,
20               4/, ' ', 10X, 'NB. D''ITER. DE QUB: ', I5,
21               5/, ' ', 10X, 'NB. CHANG. V.ECART:  ', I5)
22                RETURN
23                END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

  FLOW      14408  /XI/                              REAL          800
  FY            4  DUMMY-ARG                         REAL
  IR            7  DUMMY-ARG                         INTEGER
  LP            9  DUMMY-ARG                         INTEGER
  LPEC         10  DUMMY-ARG                         INTEGER
  NLINK         2  DUMMY-ARG UNUSED/*S*              INTEGER

 -NAME---ADDRESS--BLOCK-----PR(

  NOITE         1  DUMMY-ARG
  THETA         5  DUMMY-ARG
  TIME          6  DUMMY-ARG
  VCUM         0B  /XI/
  VN            8  DUMMY-ARG
  XOK           3  DUMMY-ARG

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

      10      26B      FORMAT          8
      15      32B      FORMAT         12
      20      37B      FORMAT         15

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  IMPRESU      5B     10

--STATISTICS--

  PROGRAM-UNIT LENGTH              117B =  79
  CM LABELLED COMMON LENGTH       3100B =  1600
  CM STORAGE USED                61300B =  25280
  COMPILE TIME                   0.145 SECONDS
```

```
      SUBROUTINE NEWTON(THETA, NLINK)
      COMMON/NET1/DIST(800), SPDO(800), WIDTH(800),
     +                CAP(800), LTYP(800)
      COMMON/XI/VCUM(800), FLOW(800)
      COMMON/COEF/ALPHA(5), BETA(5)
      COMMON WKMEM(1004)
      DIMENSION VINIT(800)
      EQUIVALENCE (VINIT(1), WKMEM(1))
      DATA MAXNEWT/10/, GAP/0.01/
*
      THETA=1.0
      DO 5 I=1, NLINK
      VINIT(I)=VCUM(I)
    5 VCUM(I)=VCUM(I)+THETA*FLOW(I)
      FNEW=SSTRAV(NLINK)
      IF(FNEW.LE.0.0) RETURN
      NITE=1
      THETA=0.0
      DO 10 I=1, NLINK
   10 VCUM(I)=VINIT(I)
   25 CALL DCOST(THETA, NLINK, F1, F2)
      OLDTETA=THETA
      THETA=THETA+F1/F2
      DO 50 I=1, NLINK
   50 VCUM(I)=VINIT(I)+THETA*FLOW(I)
      NITE=NITE+1
      IF (NITE.LE.MAXNEWT.AND.ABS(OLDTETA-THETA).GT.GAP)
     1GO TO 25
      RETURN
      END
```


--VARIABLE MAP--(LO=A)

-NAME-	-ADDRESS-	-BLOCK-	-PROPERTIES-	-TYPE-	-SIZE-
ALPHA	0B	/COEF/		REAL	5
BETA	5B	/COEF/		REAL	5
CAP	4540B	/NET1/		REAL	800
DIST	0B	/NET1/		REAL	800
FLOW	1440B	/XI/		REAL	800
FNEW	142B			REAL	
F1	145B			REAL	
F2	146B			REAL	
GAP	136B			REAL	
I	140B			INTEGER	
LTYP	6200B	/NET1/		INTEGER	800

-NAME-	-ADDRESS-	-BLOCK-
MAXNEWT	137B	
NITE	143B	
NLINK	2	DUMMY-ARG
OLDTETA	147B	
SPDO	1440B	/NET1/
THETA	1	DUMMY-ARG
VCUM	0B	/XI/
VINIT	0B	//
WIDTH	3100B	/NET1/
WKMEM	0B	//

--PROCEDURES--(LO=A)

-NAME-	-TYPE-	-ARGS-	-CLASS-
ABS	GENERIC	1	INTRINSIC
DCOST		4	SUBROUTINE
SSTRAV	REAL	1	FUNCTION

```
SUBROUTINE FOBJ        73/1     OPT=0, ROUND= A/ S/ M/-D, -DS      FTN 5.1+564          83/0
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50(
FTN5.

 1                    SUBROUTINE FOBJ(TIME,NLINK)
 2                    COMMON/NET1/DIST(800),SPDO(800),WIDTH(800),
 3                   +           CAP(800),LTYP(800)
 4                    COMMON/XI/VCUM(800),FLOW(800)
 5                    COMMON/COEF/ALPHA(5),BETA(5)
 6                    SUM=0.0
 7                    DO 100 J=1,NLINK
 8                    IF(LTYP(J).GT.1) GO TO 100
 9                    TO=60.0*DIST(J)/SPDO(J)
10                    LANE=(WIDTH(J)+2.0)/10.0
11                    CDASH=CAP(J)/LANE
12                    VO=SPDO(J)
13                    CALA=249
14                    DO 20 JJ=1,4
15                    IF(CDASH.LE.CALA)GO TO 30
16                20  CALA=CALA+250.0
17                    JJ=5
18                30  B=BETA(JJ)
19                    A=ALPHA(JJ)
20                    X=(VCUM(J)/CAP(J))**B
21                    SUM=SUM+TO*VCUM(J)*(1.0+A*X/(B+1.0))
22                100 CONTINUE
23                    PRINT 200,SUM
24                200 FORMAT(' ',10X,'F. OBJ. : ',F16.5)
25                    RETURN
26                    END


--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

 A          151B
 ALPHA        0B  /COEF/      REAL
 B          150B              REAL             5
 BETA         5B  /COEF/      REAL
 CALA       145B              REAL             5
 CAP       4540B  /NET1/      REAL
 CDASH      143B              REAL           800
 DIST         0B  /NET1/      REAL
 FLOW      1440B  /XI/        REAL           800
 J          137B              REAL           800
 JJ         146B              INTEGER
                              INTEGER

 -NAME---ADDRESS--BLOCK-----PROPERTIES

 LANE       142B
 LTYP      6200B  /NET1/
 NLINK        2   DUMMY-ARG
 SPDO      1440B  /NET1/
 SUM        136B
 TIME         1   DUMMY-ARG
 TO         141B
 VCUM         0B  /XI/
 VO         144B
 WIDTH     3100B  /NET1/
 X          152B


--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    20  INACTIVE    DO-TERM         16
    30     61B                      18
   100    102B      DO-TERM         22
   200    123B      FORMAT          24
```

[redacted] FUNCTION SSTRAV [redacted] OPT=[redacted] RO[redacted]= [redacted]6/ [redacted]0, [redacted] [redacted]TN [redacted]+5[redacted] [redacted] 83[redacted]
DB=-LONG/-UT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50
FTN5.

```
      1                 REAL FUNCTION SSTRAV(NLINK)
      2                 COMMON/NET1/DIST(800),SPDO(800),WIDTH(800),
      3                +            CAP(800),LTYP(800)
      4                 COMMON/XI/VCUM(800),FLOW(800)
      5                 COMMON/COEF/ALPHA(5),BETA(5)
      6                 SUM=0.0
      7                 DO 2000 J=1,NLINK
      8           2000  SUM=SUM+UCOST(J)*FLOW(J)
      9                 SSTRAV=SUM
     10                 RETURN
     11                 END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS--	--BLOCK-----	--PROPERTIES-------	--TYPE---------	--SIZE
ALPHA	0B	/COEF/		REAL	5
BETA	5B	/COEF/		REAL	5
CAP	4540B	/NET1/		REAL	800
DIST	0B	/NET1/		REAL	800
FLOW	1440B	/XI/		REAL	800
J	40B			INTEGER	
LTYP	6200B	/NET1/		INTEGER	800

-NAME-	--ADDRESS--	--BLOCK-----
NLINK	1	DUMMY-ARG
SPDO	1440B	/NET1/
SSTRAV	36B	
SUM	37B	
VCUM	0B	/XI/
WIDTH	3100B	/NET1/

--PROCEDURES--(LO=A)

-NAME------	--TYPE--------	--ARGS------	--CLASS-----
UCOST	REAL	1	FUNCTION

--STATEMENT LABELS--(LO=A)

-LABEL-	--ADDRESS-----	--PROPERTIES----	--DEF
2000	INACTIVE	DO-TERM	8

--ENTRY POINTS--(LO=A)

-NAME-	--ADDRESS--	--ARGS---
SSTRAV	6B	1

--STATISTICS--

PROGRAM-UNIT LENGTH	44B	= 36
CM LABELLED COMMON LENGTH	12752B	= 5610
CM STORAGE USED	61300B	= 25280
COMPILE TIME	0.109 SECONDS	

```
SUBROUTINE DCOST       73/1     OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+564          83/02
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,PL=500
FTN5.

      1                     SUBROUTINE DCOST(THETA,NLINK,F1,F2)
      2                     COMMON/NET1/DIST(800),SPDO(800),WIDTH(800),
      3                    +            CAP(800),LTYP(800)
      4                     COMMON/XI/VCUM(800),FLOW(800)
      5                     COMMON/COEF/ALPHA(5),BETA(5)
      6                     F1=F2=F3=0.0
      7                     DO 100 J=1,NLINK
      8                     IF(LTYP(J).EQ.1) THEN
      9                     A=0.0
     10                     LANE=(WIDTH(J)+2.0)/10.0
     11                     CDASH=CAP(J)/LANE
     12                     TO=60.0*DIST(J)/SPDO(J)
     13                     CALA=249
     14                     DO 20 JJ=1,4
     15                     IF(CDASH.LE.CALA)GO TO 30
     16               20    CALA=CALA+250.0
     17                     JJ=5
     18               30    B=BETA(JJ)
     19                     A=ALPHA(JJ)
     20                     CON=TO*A/(CAP(J)**B)
     21                     F1=F1-(TO+CON*(VCUM(J)**B))*FLOW(J)
     22                     F2=F2+CON*B*(VCUM(J)**(B-1.0))*(FLOW(J)**2)
     23               *     F1=F1+(TO+CON*(VCUM(J)**B))*FLOW(J)
     24               *     F2=F2+CON*B*(VCUM(J)**(B-1))*(FLOW(J)**2)
     25               *     F3=F3+CON*B*(B-1)*(VCUM(J)**(B-2))*(FLOW(J)**3)
     26               *     F1=-F1+(1.0-THETA)*F2
     27               *     F2=F2*2-(1.0-THETA)*F3
     28                     ENDIF
     29                100  CONTINUE
     30                     RETURN
     31                     END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

 A          146B                            REAL
 ALPHA        0B  /COEF/                    REAL             5
 B          155B                            REAL
 BETA         5B  /COEF/                    REAL             5
 CALA       152B                            REAL
 CAP       4540B  /NET1/                    REAL           800
 CDASH      150B                            REAL
 CON        156B                            REAL
 DIST         0B  /NET1/                    REAL           800
 FLOW      1440B  /XI/                      REAL           800
 F1           3   DUMMY-ARG                 REAL
 F2           4   DUMMY-ARG                 REAL

 -NAME---ADDRESS--BLOCK-----P
 F3         143B
 J          144B
 JJ         153B
 LANE       147B
 LTYP      6200B  /NET1/
 NLINK        2   DUMMY-ARG
 SPDO      1440B  /NET1/
 THETA        1   DUMMY-ARG
 TO         151B
 VCUM         0B  /XI/
 WIDTH     3100B  /NET1/

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    20   INACTIVE    DO-TERM        16
    30      62B                     18
   100   INACTIVE    DO-TERM        29
```

```
[redacted] FU[redacted]ION[redacted]OS[redacted] [redacted]/1 [redacted] OP[redacted], RO[redacted]= [redacted]S/ [redacted] D, [redacted] [redacted]TN [redacted]+584[redacted] [redacted] 8370[redacted]
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=50
FTN5.

     1            REAL FUNCTION UCOST( J )
     2      C
     3      C---------CALCULATE USER COST = TRAVEL TIME FOR LINK J
     4      C
     5            COMMON/NET1/DIST(800),SPDO(800),WIDTH(800),
     6           +            CAP(800),LTYP(800)
     7            COMMON/XI/VCUM(800),FLOW(800)
     8            COMMON/COEF/ALPHA(5),BETA(5)
     9            VOL=VCUM(J)
    10            GO TO(1,2,3),LTYP(J)
    11      1     A=0.0
    12            LANE=(WIDTH(J)+2.0)/10.0
    13            CDASH=CAP(J)/LANE
    14            TO=60.0*DIST(J)/SPDO(J)
    15            VO=SPDO(J)
    16            CALA=249
    17            DO 20 JJ=1,4
    18            IF(CDASH.LE.CALA) GO TO 30
    19      20    CALA=CALA+250.0
    20            JJ=5
    21      30    B=BETA(JJ)
    22            A=ALPHA(JJ)
    23            UCOST=TO*(1.0+A*(VOL/CAP(J))**B)
    24            RETURN
    25      2     UCOST=0.0
    26            RETURN
    27      3     UCOST=999.0
    28            RETURN
    29            END
```

--VARIABLE MAP--(LO=A)

-NAME---	ADDRESS--	BLOCK-----	PROPERTIES-------	TYPE---------	SIZE
A	125B			REAL	
ALPHA	0B	/COEF/		REAL	5
B	135B			REAL	
BETA	5B	/COEF/		REAL	5
CALA	132B			REAL	
CAP	4540B	/NET1/		REAL	800
CDASH	127B			REAL	
DIST	0B	/NET1/		REAL	800
FLOW	1440B	/XI/		REAL	800
J	1	DUMMY-ARG		INTEGER	

-NAME---	ADDRESS--	BLOCK-----
JJ	133B	
LANE	126B	
LTYP	6200B	/NET1/
SPDO	1440B	/NET1/
TO	130B	
UCOST	123B	
VCUM	0B	/XI/
VO	131B	
VOL	124B	
WIDTH	3100B	/NET1/

--STATEMENT LABELS--(LO=A)

-LABEL	-ADDRESS-----	PROPERTIES----	DEF
1	23B		11
2	103B		25
3	106B		27
20	INACTIVE	DO-TERM	19
30	65B		21

```
SUBROUTINE MINTREE    73/1    OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+564          83/02
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,PL=50C
FTN5.

      1                 SUBROUTINE MINTREE(NN,NCENT)
      2                 COMMON/NET/NNODE,NLINK,NOLAB(502),ND(800),NO(800)
      3                 COMMON/CHEM/LPR(502),TRAV(800)
      4                 COMMON WKMEM(1004)
      5                 DIMENSION NSET(502),TSUM(502)
      6                 EQUIVALENCE(TSUM(1),WKMEM(1)),(NSET(1),WKMEM(503))
      7           C
      8           C--------INITIALISATION
      9           C
     10                 MAX=999999
     11                 DO 10 I=1,NNODE
     12                 LPR(I)=0
     13                 TSUM(I)= 1.0E10
     14                 NSET(I)=0
     15                 IF(NOLAB(I).EQ.0)NSET(I)=MAX
     16              10 CONTINUE
     17                 DO 20 I=1,NCENT
     18              20 NSET(I)=MAX
     19           C
     20           C
     21           C
     22                 LAST=NN
     23                 N=NN
     24                 TSUM(N)=0.0
     25                 GOTO 110
     26           C
     27           C--------FETCH NEW NODE FOR SCANNING.  IF LIST EMPTY STOP
     28           C
     29             100 I=N
     30                 N=NSET(N)
     31                 IF(N.EQ.MAX)GOTO 200
     32                 NSET(I)=-1
     33           C
     34           C-------SCAN NODE N
     35           C
     36             110 I1=NOLAB(N)
     37                 K=N
     38             120 K=K+1
     39                 I2=NOLAB(K) -1
     40                 IF(I2.LT.0)GOTO 120
     41           C
     42                 DO 130 I=I1,I2
     43                 M=ND(I)
     44                 VAL=TSUM(N)+TRAV(I)
     45                 IF(VAL.GE.TSUM(M))GOTO 130
     46                 TSUM(M)=VAL
     47                 LPR(M)=I
     48           C
     49           C--------ADD M TO LIST IF FIRST TIME REACH  M  OR LIST HAS
     50           C--------ONLY ONE ELEMENT, ELSE INSERT M AT TOP LIST IF
     51           C--------M IS NOT ALREADY IN LIST
     52           C
     53                 IF(NSET(M))121,122,130
     54             121 IF(NSET(N).EQ.MAX)GOTO 122
     55                 NSET(M)=NSET(N)
     56                 NSET(N)=M
     57                 GOTO 130
     58             122 NSET(M)=MAX
     59                 NSET(LAST)=M
```

```
               LAST=M
           130 CONTINUE
      C
               GOTO 100
           200 CONTINUE
               RETURN
               END


--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS--BLOCK-----PR(

  I         163B                             INTEGER                NLINK        1B   /NET/
  I1        170B                             INTEGER                NN           1    DUMMY-ARG
  I2        172B                             INTEGER                NNODE        0B   /NET/
  K         171B                             INTEGER                ND       2430B    /NET/
  LAST      166B                             INTEGER                NOLAB        2B   /NET/
  LPR         0B   /CHEM/.                   INTEGER          502   NSET       766B   //          EQ'
  M         174B                             INTEGER                TRAV       766B   /CHEM/
  MAX       162B                             INTEGER                TSUM         0B   //          EQ'
  N         167B                             INTEGER                VAL        175B
  NCENT       2    DUMMY-ARG                 INTEGER                WKMEM        0B   //          EQ'
  ND        770B   /NET/                     INTEGER          800

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF       -LABEL-ADDRESS-----PROPERTIES----DEF       -LABEL-ADI

    10  INACTIVE   DO-TERM        16           110      67B                   36           122
    20  INACTIVE   DO-TERM        18           120      74B                   38           130
   100      57B                   29           121  INACTIVE                  54           200

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  MINTREE     5B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH             200B =  128
  CM BLANK COMMON LENGTH         1754B =  1004
  CM LABELLED COMMON LENGTH      6516B =  3406
  CM STORAGE USED               61300B =  25280
  COMPILE TIME                  0.325 SECONDS
```

```
DO=-LONG/-DT,ARG=-COMMON/-FIXED,CB= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,PL=5000
FTN5.

      SUBROUTINE PARGUB(MAXITER,GAPMIN,OMEGA,PARIMP,ISER,ISYM,LPAS)
*
*-----CRITERES D'ARRET DE GUB:
*     -----MAXITER=NOMBRE MAXIMUM D''ITERATIONS DE GUB.
*     -----GAPMIN =ARRET SI LE GAP (CX,X-Y) EST INFERIEUR
*                  OU EGAL A GAPMIN.
*
*-----PARAMETRE "OMEGA" DE LA FONCTION OBJECTIVE.
*
*-----PARAMETRE D'IMPRESSION:
*     -----PARIMP=0 IMPRIMER LES RESULTATS DE LA DERNIERE
*                   SEULEMENT
*                 1 IMPRIMER LES RESULTATS DE CHAQUE ITERATION
*                 2 IMPRIMER LES DONNEES DU RESEAU ET LES
*                   RESULTATS DE LA DERNIERE ITERATION.
*                 3 IMPRIMER LES DONNEES DU RESEAU ET LES
*                   RESULTATS DE CHAQUE ITERATION.
*
*-----ISER:  PURGE LES CONTRAINTES NON-ACTIVES(1) OU NON(0).
*
*-----ISYM:  CAS SYMETRIQUE(1) OU NON(0).
*
*-----LPAS:  LE PAS EST 1.0(1) OU NON(0).
*
      CHARACTER*80 TITRE
      INTEGER PARIMP
*
      READ 100,TITRE
  100 FORMAT(A80)
      PRINT 1000,TITRE
 1000 FORMAT('1',///,10X,A80)
*
      READ 2000,MAXITER,GAPMIN,OMEGA,PARIMP,ISER,ISYM,LPAS
 2000 FORMAT(I5,2F9.6,4I1)
      PRINT 3000,MAXITER,GAPMIN,OMEGA,PARIMP
 3000 FORMAT('0',10X,'NB. MAXIMUM D''ITERATIONS DE GUB:  ',I5,/,
     1' ',10X,'VALEUR MINIMUM DU GAP:  ',F9.6,/,' ',10X,
     2'PARAMETRE "OMEGA" DE LA FONCTION OBJ.:  ',F9.6,
     3/,' ',10X,'PARAMETRE D''IMPRESSION:  ',I1)
      PRINT 3500,ISYM,LPAS
 3500 FORMAT(' ',10X,'CAS SYMETRIQUE(1) OU NON(0)= ',I1,
     1/,' ',10X,'LE PAS EST 1.0(1) OU NON(0):  ',I1)
      IF(ISER.EQ.1)THEN
       PRINT 4000
                  ELSE
       PRINT 5000
      ENDIF
*
 4000 FORMAT('0',//,10X,'LES CONTRAINTES NON-ACTIVES SONT PURGEES')
 5000 FORMAT('0',//,10X,'LES CONTRAINTES SONT TOUTES CONSERVEES')
      RETURN
      END
```

```
      SUBROUTINE RDNET(NPOD)
*
*-----LIRE LES INFORMATIONS SUR LE RESEAU.
*
*       ----- NPOD     = NOMBRE DE PAIRES O/D.
*       ----- NNODE    = NOMBRE DE SOMMETS.
*       ----- NLINK    = NOMBRE D'ARCS.
*       ----- NOLAB(I) = CONTIENT L'ADRESSE DU PREMIER DES ARCS
*                        (X,Y)  (POUR CHACUN DES X).
*       ----- ND(I)    = L'EXTREMITE TERMINALE DE L'ARC I.
*
      COMMON /NET/NNODE,NLINK,NOLAB(502),ND(800),ND(800)
      COMMON/NET1/DIST(800),SPDO(800),WIDTH(800),
     +            CAP(800),LTYP(800)
      DIMENSION CC(2),W(2)
*
      DATA MAXNOD/502/
 2000 FORMAT (1HO,20X,'DIMENSIONS SPECIFIEES TROUVEES'/
     +                 23X,    '- SOMM',I9,I10/
     +                 23X,    '- LIEN',I9,I10/
     +                 23X,    '- NPOD',I9/ 1H1)
 2001 FORMAT (1HO,20X,'0601E LIENS NON TRIES.  LIEN: ',I4//)
*
*     LIRE DIMENSION DU RESEAU.
*
      READ 3000,NPOD,NNODE,NLINK
 3000 FORMAT (3I5)
*
*-----INITIALISATION
*
      DO 5000 I=1,MAXNOD
 5000   NOLAB(I) = 0
      NOLAB(1) = 1
      I = 0
      MNO = 0
      J = 1
 5500 I = I + 1
*
*-----LIRE INFORMATION SUR LES ARCS.
*
      READ 5600,NO(I),ND(I),DIST(I),SPDO(I),CC(1),CC(2),W(1),W(2),I1,
     1I2
 5600 FORMAT(2(I4,1X),F4.2,1X,F4.2,1X,2(F4.0,1X),2(F4.1,1X),I1,1X,I1)
      NNO=NO(I)
      IF (NNO .EQ. 0) GO TO 5900
      MNO = MAXO(NNO,ND(I),MNO)
      LTYP(I)=2
      CAP(I)=0.0
      WIDTH(I)=0.0
      IF(I2.EQ.0) GOTO 5665
      LTYP(I)=3
      GOTO 5670
 5665 IF(I1.LE.0)GO TO 5670
      CAP(I)=CC(I1)
      WIDTH(I)=W(I1)
      LTYP(I)=1
*
*-----INITIALISATION DU VECTEUR D'ADRESSE NOLAB
*
```

```
60                5670 IF (NNO-J) 5700, 5500, 5800
61                5700       PRINT 2001, I
62                           CALL EXIT
63                5800 NOLAB(NNO) = I
64                     J = NNO
65                     GO TO 5500
66                *
67                5900 I = I -1
68                     PRINT 2000, NNODE, MNO, NLINK, I, NPOD
69                     NLINK = I
70                     NNODE = MNO
71                     NOLAB(NNODE + 1) = NLINK + 1
72                     RETURN
73                     END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS--	--BLOCK-----	PROPERTIES-------	TYPE---------	SIZE
CAP	4540B	/NET1/		REAL	800
CC	255B			REAL	2
DIST	0B	/NET1/		REAL	800
I	262B			INTEGER	
I1	266B			INTEGER	
I2	267B			INTEGER	
J	265B			INTEGER	
LTYP	6200B	/NET1/		INTEGER	800
MAXNOD	261B			INTEGER	
MNO	264B			INTEGER	

-NAME-	--ADDRESS--	--BLOCK-----	PR
ND	770B	/NET/	
NLINK	1B	/NET/	
NNO	270B		
NNODE	0B	/NET/	
NO	2430B	/NET/	
NOLAB	2B	/NET/	
NPOD	1	DUMMY-ARG	
SPDO	1440B	/NET1/	
W	257B		
WIDTH	3100B	/NET1/	

--PROCEDURES--(LO=A)

-NAME------	TYPE--------	ARGS------	CLASS-----
EXIT			SUBROUTINE
MAXO	INTEGER	VAR 0	INTRINSIC

--STATEMENT LABELS--(LO=A)

-LABEL-	ADDRESS-----	PROPERTIES----	DEF	-LABEL-	ADDRESS-----	PROPERTIES----	DEF	-LABEL-	ADI
2000	154B	FORMAT	18	5500	32B		37	5700	1I
2001	170B	FORMAT	22	5600	200B	FORMAT	43	5800	
3000	176B	FORMAT	27	5665	105B		53	5900	
5000	INACTIVE	DO-TERM	32	5670	117B		60		

--ENTRY POINTS--(LO=A)

-NAME-	--ADDRESS--	ARGS---
RDNET	5B	1

[redacted]
FTN5.

```
      1                SUBROUTINE RDOD(NPOD)
      2              *
      3              *-----LIRE DONNEES POUR CHAQUE PAIRE O/D.
      4              *
      5              *      ----- NPOD      = NOMBRE DE PAIRES O/D.
      6              *      ----- LINIT(I)  = L'ORIGINE DE LA PAIRE O/D I.
      7              *      ----- LTERM(I)  = LA DESTINATION DE LA PAIRE O/D I.
      8              *      ----- DEMANDE(I)= LA DEMANDE DE LA PAIRE O/D I.
      9              *
     10                     COMMON /POD/LINIT(150),LTERM(150),DEMANDE(150)
     11              *
     12                     DO 9000 J=1,NPOD
     13                     READ 7000,LINIT(J),LTERM(J),DEMANDE(J)
     14                7000 FORMAT(2I5,F10.3)
     15                9000 CONTINUE
     16                     RETURN
     17                     END
```

```
--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE

  DEMANDE    454B  /POD/                      REAL           150
  J           54B                             INTEGER
  LINIT        0B  /POD/                      INTEGER        150
  LTERM      226B  /POD/                      INTEGER        150
  NPOD         1   DUMMY-ARG                  INTEGER

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

   7000      40B     FORMAT          14
   9000   INACTIVE   DO-TERM         15

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  RDOD          5B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH              60B =  48
  CM LABELLED COMMON LENGTH       702B =  450
  CM STORAGE USED               61300B =  25280
  COMPILE TIME                  0.099 SECONDS
```

```
SUBROUTINE IMPNET      73/1     OPT=0, ROUND= A/ S/ M/-D, -DS      FTN 5.1+564          83/02/
DO=-LONG/-OT, ARG=-COMMON/-FIXED, CS= USER/-FIXED, DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST, PL=5000
FTN5.

      1                 SUBROUTINE IMPNET(PARIMP)
      2           *
      3           *-----IMPRESSION DES DONNEES DU RESEAU:
      4           *     - LES ARCS (X,Y) DU RESEAU.
      5           *     - LES FONCTIONS DE COUT ASSOCIEES AUX ARCS.
      6           *
      7                 COMMON/NET/NNODE, NLINK, NOLAB(502), ND(800), NO(800)
      8                 COMMON/NET1/DIST(800), SPDO(800), WIDTH(800),
      9                +           CAP(800), LTYP(800)
     10                 INTEGER PARIMP
     11                 PRINT 50
     12            50   FORMAT('1', //, 15X, 'LE RESEAU COMPREND LES ARCS SUIVANTS: ', ///)
     13           *
     14                 DO 200 I=1, NLINK
     15                 PRINT 100, NO(I), ND(I)
     16           100   FORMAT(' ', 25X, '(', I5, ', ', I5, ')')
     17           200   CONTINUE
     18           *
     19           *-----IMPRESSION DES PARAMETRES DES FONCTIONS DE COUTS.
     20           *
     21                 PRINT 500
     22           500   FORMAT('0', 5X, 'IMPRESSION DES PARAMETRES DES FONCTIONS DE COUTS',
     23                1//)
     24                 PRINT 700
     25           700   FORMAT('0', 5X, 'I', 5X, 'DIST(I)', 5X, 'SPDO(I)', 5X, 'WIDTH(I)', 4X,
     26                1'CAP(I)', 6X, 'LTYP(I)')
     27                 DO 800 I=1, NLINK
     28                 PRINT 750, I, DIST(I), SPDO(I), WIDTH(I), CAP(I), LTYP(I)
     29           750   FORMAT(' ', 3X, I3, 4F12.5, 6X, I3)
     30           800   CONTINUE
     31                 RETURN
     32                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS--BLOCK-----PROPERTIES-------TYPE---------SIZE      -NAME---ADDRESS--BLOCK-----PR(

 CAP        4540B  /NET1/                    REAL             800    NNODE        0B  /NET/
 DIST          0B  /NET1/                    REAL             800    NO        2430B  /NET/
 I           171B                           INTEGER                 NOLAB        2B  /NET/
 LTYP       6200B  /NET1/                    INTEGER          800    PARIMP       1   DUMMY-ARG UN(
 ND          770B  /NET/                     INTEGER          800    SPDO      1440B  /NET1/
 NLINK         1B  /NET/                     INTEGER                 WIDTH     3100B  /NET1/

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF      -LABEL-ADDRESS-----PROPERTIES----DEF

     50      100B      FORMAT          12      700      123B      FORMAT          25
    100      107B      FORMAT          16      750      134B      FORMAT          29
    200   INACTIVE     DO-TERM         17      800   INACTIVE     DO-TERM         30
    500      113B      FORMAT          22
```

```
      SUBROUTINE IMPOD(NPOD)
*
*-----IMPRESSION DES DONNEES POUR CHAQUE PAIRE O/D.
*
      COMMON /POD/LINIT(150),LTERM(150),DEMANDE(150)
      PRINT 8000
 8000 FORMAT('1',//,10X,'ORIGINE DESTINATION     DEMANDE')
      DO 9000 J=1,NPOD
 9000 PRINT 9500,LINIT(J),LTERM(J),DEMANDE(J)
 9500 FORMAT('0',11X,I5,5X,I5,4X,F10.5)
      PRINT 9800
 9800 FORMAT('1',//,20X,'IMPRESSION DES RESULTATS',///)
      RETURN
      END
```

--VARIABLE MAP--(LO=A)

-NAME---	ADDRESS--	BLOCK-----	PROPERTIES-------	TYPE---------	SIZE
DEMANDE	454B	/POD/		REAL	150
J	101B			INTEGER	
LINIT	0B	/POD/		INTEGER	150
LTERM	226B	/POD/		INTEGER	150
NPOD	1	DUMMY-ARG		INTEGER	

--STATEMENT LABELS--(LO=A)

-LABEL-	ADDRESS-----	PROPERTIES----	DEF
8000	43B	FORMAT	7
9000	INACTIVE	DO-TERM	9
9500	51B	FORMAT	10
9800	55B	FORMAT	12

--ENTRY POINTS--(LO=A)

-NAME---	ADDRESS--	ARGS---
IMPOD	5B	1

--STATISTICS--

```
PROGRAM-UNIT LENGTH              105B =   69
CM LABELLED COMMON LENGTH        702B =  450
CM STORAGE USED                61300B = 25280
COMPILE TIME                   0.115 SECONDS
```

BIBLIOGRAPHIE

Aashtiani, H.Z. "The Multi-Modal Traffic Assignment Problem." Thèse de doctorat présentée à Massachusetts Institute of Technology, mai 1979.

Aashtiani, H.Z. and Magnanti, T.L. "Equilibria on a Congested Transportation Network." SIAM Journal on Algebraic and Discrete Methods, 2 (1980), 213-226.

Auslender, A. "Problèmes de Minimax via l'analyse convexe et les inégalités variationnelles: théorie et algorithmes." Lecture Notes in Economics and Mathematical Systems, Edited by M. Beckmann, Providence; G. Goos, Karlsruhe, and H.P. Kiinzi, Zurich; Springer-Verlag, Berlin; Heidelberg, New-York, 1972.

Auslender, A. Optimisation. Méthode numérique. Paris: Masson, 1976.

Auslender, A. Recherche des points de selle d'une fonction. Faculté des Sciences de Clermont-Ferrand.

Avriel, M. Nonlinear Programming: Analysis and Methods. Prentice Hall, 1976.

Bakushinskii, A.B.; Poljak, B.T. "On the Solution of Variational Inequalities." Dan. 219, 5 (1974), 1038-1041; Soviet Math. Doklady, 15 (1974), 1705-1710.

Beckmann, M.; McGuire, C.; Winsten, C.B. Studies in Economics of Transportation. New Haven, Connecticut: Yale University Press, 1956.

Bénichou, M.; Gauthier, J.M.; Hentges, G.; Ribiere, G. "The Efficient Solution of Large-Scale Linear Programming Problems-some Algorithmic Techniques and Computational Results." Mathematical Programming, 13 (1977), 280-322.

Bensoussan, A.; Lions, J.L.; Teman, R. "Sur les méthodes de décomposition, de décentralisation et de coordinations et d'applications." Cahiers de l'IRIA, 11 (1972), 5-189.

Berge, C. Topological Spaces. London: Oliver Boyd, 1963.

Bertsekas, D.P. "On the Goldstein-Levitin-Polyak Gradient Projection Method." IEEE Transactions on Automatics Control, Vol. AC-21, 2 (1974).

Bertsekas, D.; Gafni, E. Projection Methods for Variational Inequalities with Application to the Traffic Assignment Problem. Technical Report, Laboratory for Information and Decision Systems, MIT, Cambridge, MA(1980).

Blankenship, J.W.; Falk, J.E. "Infinitely Constrained Optimization Problems." Journal of Optimization Theory and Applications, 19, 2 (1976), 261-281.

Branston, D.; Florian, M.; James, L.; Nguyen, S.; Trahan, M.; Développement d'un outil d'étude pour déterminer l'impact de nouveaux aménagements sur la circulation routière de l'île de Hull. Contrat de recherche exécuté pour le service des projets spéciaux et du développement. Ministère des Transports du Québec. Université de Montréal, Publication #46, Centre de recherche sur les transports, (septembre 1975).

Butz, A.R. "Iterative Saddle Point Techniques." SIAM Journal on Applied Mathematics, 15, 3 (1967).

Canon, M.D.; Cullum, C.D. "A Tight Upper Bound on the Rate of Convergence of the Frank-Wolfe Algorithm." SIAM Journal on Control, 6, 4 (1968), 509-516.

Cheney, E.W.; Goldstein, A.A. "Newton's Method of Convex Programming and Tchebycheff Approximation." Numerische Mathematik, 253-268.

Cohen, G. "Auxiliary Problem Principle & Decomposition of Optimization Problems." Journal of Optimization Theory and Applications, 32, 3 (1980), 277-305.

Dafermos, S.C. "An Extended Traffic Assignment Model with Applications to Two-Way Traffic." Transportation Science, 5 (1971), 366-389.

Dafermos, S.C. "The Traffic Assignment Problem for Multiclass User Transportation Networks." Transportation Science, 6 (1972), 73-87.

Dafermos, S. "A Multicriteria Route-Mode Choice Traffic Equilibrium Model." Lafschetz Center for Dynamical Systems, Brown University, Providence, Rhode Island, 02912.

Dafermos, S. "Traffic Equilibrium and Variational Inequalities." Transportation Science, 14, 1 (1980a).

Dafermos, S. Decomposition Algorithms for the General Asymetric Traffic Equilibrium Problem. Lefschetz Center for Dynamical Systems, Brown University, Providence, R.I., (1980b).

Dafermos, S.C. "The General Multimodal Network Equilibrium Problem With Elastic Demand." Networks, (1980c).

Dafermos, S.C.; Sparrow, F.T. "The Traffic Assignment Problem for a General Network." JOURNAL OF RESEARCH of National Bureau of Standard-B. Mathematical Sciences, 73b, 2 (April-June 1969), 91-118.

Dafermos, S. "An Iterative Scheme for Variational Inequalities." Mathematical Programming, 26 (1983), 40-47.

Dafermos, S. "Relaxation Algorithms for the General Asymmetric Traffic Equilibrium Problem." Transportation Science, 16, 2 (1982).

Danskin, John M. The Theory of Max-Min and its Applications to Weapons Allocations Problems. Springer-Verlag, New York Inc., Econometrics and Operations Research V, 1967.

Dantzig, G.B. Linear Programming and Extensions. Princeton, N.J.: Princetown University Press, 1963.

Dantzig, G.B.; Van Slyke, R.M. "Generalized Upper Bounding Techniques." Journal of Computer and System Sciences, 1 (1967), 213-226.

Dem'yanov, V.F. "On the Solution of Several Minimax Problems 1." Kibernetika, 2, 6 (1966), 58-66, (Cybernetics, 2, 6 (1966), 47-53).

Dem'yanov, V.F.; Malozemov, V.N. Introduction to MINIMAX. 1974.

Dem'yanov, V.F.; Pevnyi, A.B. "Numerical Methods for finding Saddle Points." Z. Vycisl. Mai Imat. Fiz., 12, (1972) 1099-1127. U.S.S.R. Computational Mathematics and Mathematical Physics, 12, 5 (1972), 11-52.

Dial, R.; Glover, F.; Karney, D.; Klingman, D. "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shorthest Path Trees." Networks, 9, 3 (1979), 215-250.

Dinkel, John J.; Elliott, William H.; Hochenberger, Gary A. "Computational Aspects of Cutting-Plane Algorithms for Geometric Programming Problems." Mathematical Programming, 13 (1977), 200-220.

Dunn, J.C. "Global and Asymptotic Convergence Rate Estimates for a Class of Projected Gradient Processes." SIAM Journal Control and Optimization, 19, 3 (1981), 368-400.

Dupuis, C. Une application de la technique d'approximation linéaire partielle en programmation convexe. Université de Montréal, Centre de recherche sur les transports, Publication #216, décembre 1980.

Eaves, B.C.; Saigal, R. "Homotopies for Computation of Fixed Points on Unbounded Regions. Mathematical Programming, 3 (1972), 225-237.

Eaves, C.B.; Zangwill, W.I. "Generalized Cutting Plane Algorithms." SIAM Journal Control, 9, 4 (1971), 529-542.

Elzinga, J.; Moore, T.G. Computational Experience with the Central Cutting Plane Algorithm. Proceedings of ACM Annual Conference, Atlanta, Georgia, (1973), 451-455.

Elzinga, J.; Moore, Thomas, G. "A Central Cutting Plane Algorithm for the Convex Programming Problem." Mathematical Programming, 8 (1975), 134-145.

Erlander, S.; Nguyen, S.; Stewart, N.F. "On the Calibration of the Combined Distribution-Assignment Model." Transportation Research B, 13B (1979), 259-267.

Ermolev, Yu.M. "Methods of Solution of Nonlinear Extremal Problems." Kibernetica, 2, 4 (1966), 1-17.

Fang, S.C. "Fixed Points Models for the Equilibrium Problems on Transportation Networks." Mathematics Research, Report no 80-7, department of Mathematics, University of Mariland, Baltimore Country, (1980).

Ferland, J.A. Programmation linéaire: théorie et illustrations. Notes de cours, Département d'informatique et de recherche opérationnelle, 1979.

Ferland, J.A. Programmation non-linéaire. Notes de cours, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1979.

Fichefet, J. Quelques classes de problèmes Max-Min. Cahiers du Centre d'Etudes de Recherche Opérationnelle, 18 (1976), 13-47.

Fisk, C.S.; Boyce, D.E. A Perspective on the Problem of Combining Network Equilibrium with Spatial Interaction and Activity Models. Publication #1, Transportation Planning Group, Department of Civil Engineering, University of Illinois, 1981.

Fisk, C.; Nguyen, S. A Unified Approach for the Solution of Network Equilibrium Problems. Publication #169, Centre de recherche sur les transports, Université de Montréal, 1980a.

Fisk, C.; Nguyen, S. Solution Algorithms for Network Equilibrium Models with Asymmetric User Costs. Publication #167, Centre de recherche sur les transports, Université de Montréal, 1980b.

Florian, M.; Spiess, H. The Convergence of Diagonalization Algorithms for Asymmetric Equilibrium Problems. Transportation Research B 16B (1982), 477-483.

Florian, M.; Guélat, J.; Spiess, H. An Efficient Implementation of the PARTAN Variant of the Linear Approximation Method for the Network Equilibrium Problem. A paraître (1983).

Florian, M.; Los, M. A New Look at Static Spatial Price Equilibrium Models (Reformulation and Solution Algorithms for the Samuelson, Takayama-Judge Model and Extensions). Publication #196, Centre de recherche sur les transports, Université de Montréal, 1981.

Florian, M.; Nguyen, S.; Pallotino, S. A Dual Simplex Algorithm for Finding all Shortest Paths. Publication #139, Centre de recherche sur les transports, Université de Montréal, 1979.

Florian, M. An Improved Linear Approximation Algorithm for the Network Equilibrium (Packet Switching) Problem. Publication #57, Centre de recherche sur les transports, Université de Montréal, 1977.

Florian, M. "A Traffic Equilibrium Model of Travel by Car and Public Transit Modes." Transportation Science, 11 (1977), 166-179.

Florian, M. "The Convergence of Diagonalisation Algorithms for Fixed Demand Asymmetric Network Equilibrium Problems. Publication #198, Centre de recherche sur les transports, Université de Montréal, 1981.

Florian, M.; Nguyen, S. "A Method for Computing Network Equilibrium with Elastic Demands." Transportation Science, 8, 4 (1974).

Florian, M.; Nguyen, S.; Ferland, J. "On the Combined Distribution-Assignment of Traffic." Transportation Science, 9, 1 (1975).

Florian, M.; Nguyen, S.; Pallotino, S. A Dual Simplex Algorithm for Finding all Shortest Paths. Publication #139, Centre de recherche sur les transports Université de Montréal, 1979.

Ford, L.R., Jr.; Fulkerson, D.R. Flows in Networks. Princeton University Press, Princetown, New Jersey, 1962.

Frank, M.; Wolfe, P. "An Algorithm for Quadratic Programming." Naval Research Logistics Quarterly, 3 (1956), 95-110.

Gallant, P. "Méthodes de résolution du problème d'affectation du trafic avec demande variable." Mémoire de maîtrise présenté au département d'informatique et de recherche opérationnelle, Université de Montréal, 1979.

Gartner, N.H. "Optimal Traffic Assignment with Elastic Demands: A Review Part I. Analysis Framework." Transportation Science, 14, 2 (1980).

Gartner, N.H. "Optimal Traffic Assignment with Elastic Demands: A Review Part II. Algorithmic Approaches." Transportation Science, 14, 2 (1980).

Geoffrion, A.M. "Generalized Benders Decomposition. Journal of Optimization Theory and Applications, 10, 4 (1972).

Glowinski, G.; Lions, J.L.; Trémolières, R. Analyse numérique des inéquations variationnelles. Méthodes mathématiques de l'informatique. Bordas, Paris, (1976), 13-14.

Gochet, W.; Smeers, Y. On the Use of Linear Programs to Solve Prototype Geometric Programs. CORE, Discussion Paper, no 7229, 1972.

Gonzaga, C.; Polak, E. "On Constraint Dropping Schemes and Optimality Functions for a Class of Outer Approximations Algorithms." SIAM Journal Control and Optimization, 17, 4 (1979)

Gribik, P.R. A Central-Cutting-Plane Algorithm for Semi-Infinite Programming Problems. Lecture Notes in Control and Information Sciences, #15, Edited by A.V. Balakrishnan and M. Thoma, Semi-Infini Programming, (1978), 66-82.

Gribik, P.R.; Kortanek, K.O. "Equivalence Theorems and Cutting Plane Algorithms for a Class of Experimental Design Problems." SIAM Journal on Applied Mathematics, 32, 1 (1977), 232-259.

Guélat, J. Algorithmes pour le problème d'affectation du trafic avec demandes fixes - Comparaisons. Rapport technique, Centre de recherche sur les transports, Université de Montréal, 1982.

Gustafson, S.A. "Un algorithme à trois phases destiné à résoudre les programmes semi-infinis non-linéaires." Operational Research, (1981), 495-508.

Gustafson, S.A.; Kortanek, K.O. Computational Schemes for Semi-infinite Programs. Technical Report Department of Mathematics, Carnegie-Mellon University, Pittsburgh, PA (a).

Gustafson, S.A.; Kortanek, K.O. "Numerical Treatment of a Class of Semi-Infinite Programming Problems," Naval Research Logistics Quarterly, 20 (1973), 477-504, (b).

Hartley, H.O.; Hocking, R.R. "Convex Programming by Tangential Approximation." Management Sciences, 9, 4 (1963), 600-612.

Hearn, D.W. "The Gap Function of a Convex Program." Operations Research Letters, 1, 2 (1982), 67-71.

Hearn, D.W.*; Nguyen, S.**; Dual and Saddle Functions Related to the Gap Function. Research Report No 82-4, *Industrial & Systems Engineering Department of Florida, Gaines-Ville, Fl 32611; **Université de Montréal, Canada, 1982.

Hearn, D.W.; Nguyen, S.; Lawphongpanich, S. "Convex Programming Formulations of the Asymmetric Traffic Assignment Problem." Research Report, no 83-12, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611, (1983).

Hettich, R. "A Comparaison of Some Numerical Methods for Semi-infinite Programming." Lecture Notes in Control and Information Sciences, Edited by A.V. Balakrishnan and M. Thoma, Semi-Infinite Programming, Proceedings of a Workshop Bad Honnef (August 30-September 1, 1978), 112-125.

Ho, J.K.; Loute, E. "An Advanced Implementation of the Dantzig-Wolfe Decomposition Algorithm for Linear Programming.", Mathematical Programming, 20 (1981). 303-326.

Holloway, C.A. "An Extension of the Frank and Wolfe Method for Feasible Directions." Mathematical Programming, 6 (1974), 14-27.

Honstede, W. Van "An approximation method for Semi-Infinite Problems." Lecture Notes in Control and Information Sciences, Edited by A.V. Balakrishnan and M. Thoma, Semi-Infinite Programming, Proceedings of a Workshop Bad Honnef, (August 30-September 1, 1978), 126-136.

Huard, P. "Tour d'horizon: programmation non linéaire." Revue Française d'informatique et de recherche opérationnelle No R-1, (1971), 3-48.

Jewell, W.S. "Models for Traffic Assignment." Transportation Research, 1 (1967), 31-46.

Kakutani, S. "A Generalization of a Brouwer's Fixed Point Theorem." Duke Mathematics Journal, 8 (1941), 457-459.

Karmanov, V. Programmation Mathématique. Moscou: Editions MIR, 1977.

Kelley, J.E. Jr. "The Cutting Plane Method for Solving Convex Programs." SIAM Journal on Applied Mathematics, 8 (1960), 703-712.

Kinderlehrer, D.; Stampacchia, G. An Introduction to Variational Inequalities and their Applications. Academic Press Inc., 1980.

Kluge, R.; Telschow, G. On the Convergence and Speed of Convergence of Some Iteration Methods for Variational Inequalities (I), Theory of Nonlinear Operators. Edited by Kluge and Müller, Akademe-Verlag, Berlin, (1977), 135-163.

Kuhn, H.W.; Tucker, A.W. "Nonlinear Programming." Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. Edité par J. Neyman, University of California Press, 1961, 481-492.

Lasdon, Leon S. Optimization Theory for Large Systems. Edited by Macmillan Publishing Co., Inc., New York, (1970).

Laurent, P.J.; Carasso, C. "An Algorithm of Successive Minimization in Convex Programming." Exposé présenté au IX International Symposium on Mathematical Programming, Budapest, (1976).

Lawphongpanich, S.; Hearn, D.W. "Simplicial Decomposition of the Asymmetric Traffic Assignment Problem." Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida 32611.

Leblanc, L.J.; Morlok, E.K.; Pierskalla, W.P. "An efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. Transportation Research, 9 (1975), 309-318.

Lemaire, B. "Problèmes MIN-MAX et applications au contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles linéaires." Thèse de doctorat, Université de Montpellier, Secrétariat de Mathématiques, Publication #89, Octobre 1970.

Lesslie-Jeffery, Catherine. "Le psychologue en matière de transport; cet ami inconnu des voyageurs." TRANSPO, 2, 2 (1979).

Leventhal, T.L.; Nemhauser, G.L.; Trotter, L.E. "A Column Generation Algorithm for Optimal Traffic Assignment," Transportation Science, 7 (1973), 168-176.

Levitin, E.S.; Polyak, B.T. "Constrained Minimization Methods." Zh. Vychisl. Mat. Mat. Fiz. (in Russian), 6 (1966), 787-823, Also, U.S.S.R. Computational Mathematics and Mathematical Physics, (in English), 6 (1966), 1-50.

Levitin, E.S. "A General Minimization Method for Unsmooth Extremal Problems." Zh. Vȳchisl. Mat. Mat. Fiz., 9 , 4 (1969), 783-806.

Luenberger, David G. Optimization by Vector Space Methods. John Wiley & Sons Inc., 1969.

Luenberger, David G. Introduction to Linear and Nonlinear Programming. Standford University, Addison-Wesley Publishing Company, 1973.

Magnanti, T.L.; Wong, R.T. "Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria." Operations Research, 29, 3 (1981) 464-484.

Magnanti, T.L. Models and Algorithms for Predicting Urban Traffic Equilibrium. A paraître dans Transportation Planning Models, M. Florian, éditeur, North-Holland (1982).

Majstrovski, G.D. "On Gradient Method for Saddle Points Searching." EMM, 12, 5 (1976), 917-929, (en russe).

Mancino, O.G.; Stampacchia, G. "Convex Programming and Variational Inequalities." Journal of Optimization Theory and Applications, 9, 1 (1972).

Mangasarian, O.L.; Ponstein, J. "MINIMAX and Duality in Nonlinear Programming." Journal Mathematical Analysis and Applications, 11 (1965), 504-518.

Mangasarian, O.L. "Pseudo-Convex Functions." SIAM Journal Control Ser. A, 3, 2 (1965).

Marcotte P. A New Algorithm for Solving Variational Inequalities Over Polyhedra, With Application to the Traffic Assignment Problem. Publication #316, Centre de recherche sur les transports, Université de Montréal, Septembre 1983.

Martinet, B. "Régularisation d'inéquations variationnelles par approximations successives." Revue Française d'Informatique et de recherche opérationnelle, R-3 (1970), 154-159.

Martinet, B. "Algorithmes pour la résolution de problèmes d'optimisation et de Minimax." Thèse présentée à l'Université Scientifique et Médical de Grenoble, 1972.

McCormick, G.P.; Tapia, R.A. The Gradient Projection Method and Curry's Theorem. MRC Technical Summary Report #1074, University of Wisconsin, 1970.

McCormick, G.P.; Tapia, R.A. "The Gradient Projection Method Under Mild Differentiability Conditions." SIAM Journal Control, 10, 1 (1972).

Merrill, O.H. Applications and Extensions of an Algorithm that Computes Fixed Points of Certain Non-Empty Convex Upper Semi-Continuous Point to Set Mappings. Technical Report #71-7, Department of Industrial Engineering, University of Michigan, Ann Arbor, 1971.

Ministère des Transports. Les voies de l'avenir, la recherche et le développement en transport. Publié à la Direction générale des publications du Ministère des Communications du Québec, (1983).

Minty, G.J. "On the Monotonicity of the Gradient of a Convex Function." Pacific Journal of Mathematics, 14, (1964), 243-247.

Nguyen, S. "An Algorithm for the Traffic Assignment Problem." Transportation Science, 8 (1974), 203-216.

Nguyen, S. "A Unified Approach to Equilibrium Methods of Traffic Assignment." Traffic Equilibrium Methods, M. Florian Editeur, vol. 118, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, Berlin, 1976.

Nguyen, S.; Dupuis, C. "An Efficient Method for Computing Traffic Equilibria in Networks with Asymmetric Transportation Costs." C.R.T. Report #205-A, University of Montreal, September 1981.

Nguyen, S.; Lefebvre, L.J. TRAFFIC ANSI An Equilibrium Traffic Assignment Program (ANSI FORTRAN version). Publication #220, Centre de recherche sur les transports, Université de Montréal, 1981.

Ortega, J.M.; Rheinboldt, W.C. Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York, 1970.

Pallotino, S. Adaptation de l'algorithme de D'esopo-Pape pour la détermination de tous les chemins les plus courts: améliorations et simplifications Publication #136, Centre de recherche sur les transports, Université de Montréal, 1979.

Pang, J.S*; Yu, C.S** "Linearized Simplicial Decomposition Methods for Computing Traffic Equilibria on Networks." *Schhol of Management and Administration, The University of Texas at Dallas, Box 688, Richardson, Texas 75080 U.S.A. **Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, U.S.A., (1982).

Pang, J.S.; Chan, D. "Iterative Methods for Variational and Complementarity Problems." Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A., (1981). Aussi dans Mathematical Programming, 24, 3 (1982), 284-313.

Parikh, S.C. "Approximate Cutting Planes in Nonlinear Programming." Mathematical Programming, 11 (1976), 194-198.

Polyak, B.T. "A General Method for Solving Extremal Problems." Doklady Akademija Nauk SSR, 174, 1 (1967), 33-36.

Polyak, B.T. "Minimization of Unsmooth Functionals." Zh. Vychisl Mat. Mat. Fiz. 9, 3 (1969), 509-521.

Polyak, B.T. "Subgradient Methods: A Survey of Soviet Research."

Polyak, B.T. "Convergence of Methods of Feasible Directions in Extremal Problems." Zh. Vychisl. Mat. Mat. Fiz., 11, 4 (1971), 855-869.

Potts, R.B.; Oliver, R.M. Flows in Transportation Networks. Academic Press, 1972.

Powell, S. Row Generation Methods. University of Kent, Canterbury, U.K.

Powell, W.B.; Sheffi, Y. "The convergence of Equilibrium Algorithms with Predetermined Step Sizes." Civil Engineering Department Massachusetts Institute of Technology, Cambridge, MA 02139, (1979).

Project EMME. Equilibre Multimodal/Multimodal Equilibrium. Summary Report, Publication #101, Centre de recherche sur les transports, Université de Montréal, juin 1978.

Rapport du comité des transports de la région de Montréal. Le transports des personnes. Septembre 1977.

Recherches transport. Le rôle de l'automobile au Québec. Transports Québec, 1982.

Rockafellar, R.T. "Monotone Operators Associated with Saddle-Functions and Minimax Problems." Proceedings Symposium in Pure Math., 18 (1970), 241-250.

Rockafellar, R.T. "Monotone Operators and the Proximal Point Algorithm." SIAM Journal Control and Optimization, 14, 5 (1976).

Rockafellar, R.T. "A General Correspondance Between Dual Minimax Problems and Convex Programs." Pacific Journal of Mathematics, 25 (1968), 597-612.

Rockafellar, R.T. Convex Analysis. Princeton, New-Jersey, Second Printing, 1972.

Rockafellar, R.T. Conjugated Duality and Optimization. Society for Industrial and Applied Mathematics, 1974.

Rockafellar, R.T. Monotone Operators and Augmented Lagrangian Methods in Nonlinear Programming. Nonlinear Programming 3, Edited by O.L. Mangasarian, R.R. Meyer, S.M. Robinson, Academic Press, (1978), 1-25.

Shaigetz, M. Amélioration de la convergence de l'algorithme de Frank et Wolfe appliqué au problème d'affectation de trafic avec demandes fixes. Publication #150, Centre de recherche sur les transports, Université de Montréal novembre 1978.

Shapiro, J.F. Mathematical Programming: Structures and Algorithms, A Wiley-Interscience Publication, John Wiley & Sons, 1979.

Sharp, J.F.; Wagner, M.H. "Generalized Programming by linear approximation of the Dual Gradient: Equality Constraints." Naval Research Logistics Quarterly, 22, 4 (1975), 145-153.

Shepilov, M.A. "The Generalized Gradient Method for Extremal Problems." Zh. Vychisl. Mat. Mat. Fiz., 16, 1 (1976), 242-247.

Shor, N.Z. "The Rate of Convergence of the Generalized Gradient Descent Method." Kibern, 4, 3, 98-99; Cybernetics, 4, 3 (1968), 79-80.

Shu-Cherng, Fang. Trafic Equilibria on Multiclass-User Transportation Networks Analysed via Variational Inequalities. Department of Mathematics, University of Maryland, Baltimore County, Maryland 21228.

Sibony, M. "Méthodes itératives pour les équations et inéquations aux dérivées partielles nonlinéaires de type monotone." Calcolo, 7 (1970), 65-183.

Smith, M.J. "The Existence, Uniqueness and Stability of Traffic Equilibria." Transportation Research, 13B (1979), 295-304.

Smith, M.J. "The Existence and Calculation of Traffic Equilibria" Transportation Research, (1983a).

Smith, M.J. "An Algorithm for Solving Asymmetric Equilibrium Problems with a Continuous Cost-Flow Function." Transportation Research, (1983b).

Smith, M.J. "Junction Interactions and Monotonicity in Traffic Assignment." Department of Mathematics, University of York, Heslington, York, YO15DD United Kingdom, (1979).

Smith, M.J. "The Existence of a Solution to the Traffic Assignment Problem When There are Junction Interactions." Department of Mathematics, University of York, Heslington, York, YO15DD United Kingdom, (1979).

Smith, M.J. "The effect of Monotonicity on the Set of Solutions to the Traffic Assignment Problem." Department of Mathematics, University of York, Heslington, York, YO15DD United Kingdom, (1979).

Steenbrink, P.A. Optimization of Transportation Networks. New York, Wiley, 1974.

Strobel, H. Computer Controlled Urban Transportation, A Survey of Concepts, Methods, and International Experiences. International Series on Applied Systems Analysis, John Wiley & Sons, 1982.

Szymanowski, J.; Ruszczynski, A. An Accuracy Selection Algorithm For the Modified Gradient Projection Methods in Minimax Problems. Institute of Automatic Control, Technical University of Warsawa 00-665, Warsawa, Poland.

Todd, M.J. "The Computation of Fixed Points and Applications." Lecture Notes in Economics and Mathematical Systems, #124, Springer-Verlag, New York, (1976).

Topkis, D.M. "Cutting-Plane Methods Without Nested Constraint Sets." Operations Research, 18 (1970), 404-413.

Topkis, D.M. "A Note on Cutting Plane Methods Without Nested Constraint Sets." ORC, 69-36, Operations Research Center, College of Engineering, Berkeley, California, (1969).

Twidale, P. "Le temps c'est de l'argent." TRANSPO 81, 4, 4 (1981).

Veinott, A.F.Jr. "The Supporting Hyperplane Method for Unimodal Programming." Operations Research, 15 (1967), 147-152.

Wagner, M.H.; Sharp, J.F. "Generalized Programming by Linear Approximation of the Dual Gradient: Convex Programming Case." Management Science, 23, 12 (1977), 1307-1313.

Wardrop, J.G. "Some Theoretical Aspects of Road Traffic Research" Proc. Inst. Engineers, Part II, (1952), 325-378.

Watson, G.A. "A Multiple Exchange Algorithm for Multivariate Chebyshev Approximation." SIAM Journal on Numerical Analysis, 12 (1975), 46-52.

Wolfe, P. "Convergence Theory in Nonlinear Programming." Integer and Nonlinear Programming, Edited by J.A. Abadie, North-Holland, (1970), 1-36.

Zangwill, W.I. Nonlinear Programming: A Unified Approach. Prentice Hall, 1969.

Zoutendijk, G. Methods of Feasible Directions. Elsevier, Amsterdam, Netherlands, 1960.

Zuhovickii, S.; Polyak, R.A.; Primak, M.E. "Two Methods of Search of Equilibrium of N person-concave Games." Soviet Mathematics-Doklady, 10, 2 (1969).

Zukhovitskii, S.I.; Polyak, R.A.; Primak, M.E. "Concave Multiperson Games. Numerical Methods." Ekonomika i matematicheski metody, (1971).